SNOWFLAKE MASTERY

Easy steps to gain expertise

BY

ALPHAEDGE

SOLUTIONS

Snowflake - **<u>Database built in the cloud for the cloud</u>**

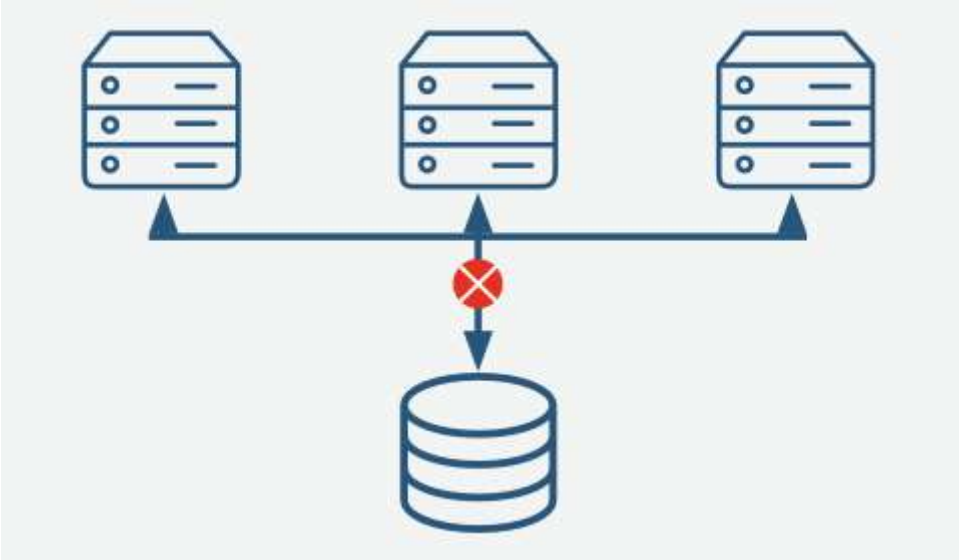
1. Founded in 2012
2. As of today 250PB data under management
3. Snowflake runs on following cloud services
    a. AWS - 2014
    b. Azure - 2018
    c. GCP - 2020

Snowflake is

1. Simple - Software as a service, plug and pay.
2. Scalable - data, compute ~ users/loads
3. Elasticity - scale up, down
4. Cost effective - pay for what you use
5. Data diversity - can store different types of data
    ~structured/semi-structured

ALPHAEDGE
SOLUTIONS

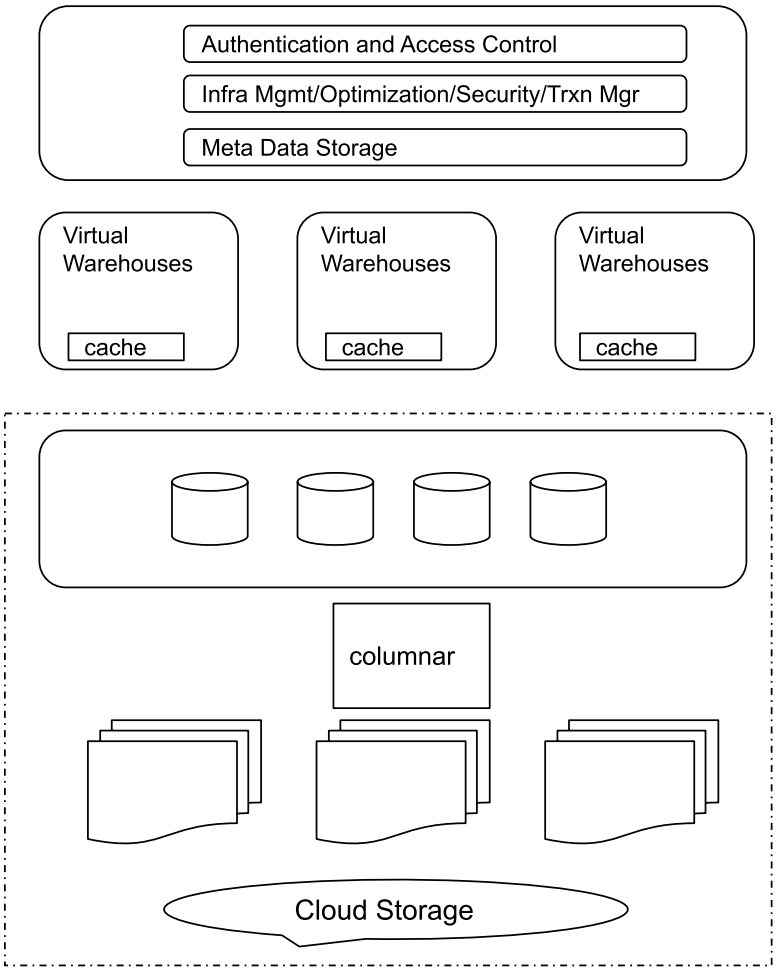
Before we talk about Snowflake Architecture ..lets see the problem in traditional architectures.



Limitation of Shared disk-architecture ~ concurrent queries

Limitation of non shared disk architecture ~ need to shuffle data across nodes as nodes increased

Snowflake Architecture



Hybrid of shared disk and non-shared disk

SERVICE LAYER

Brain of the system

QUERY PROCESSING

Muscle of the system
Query execution

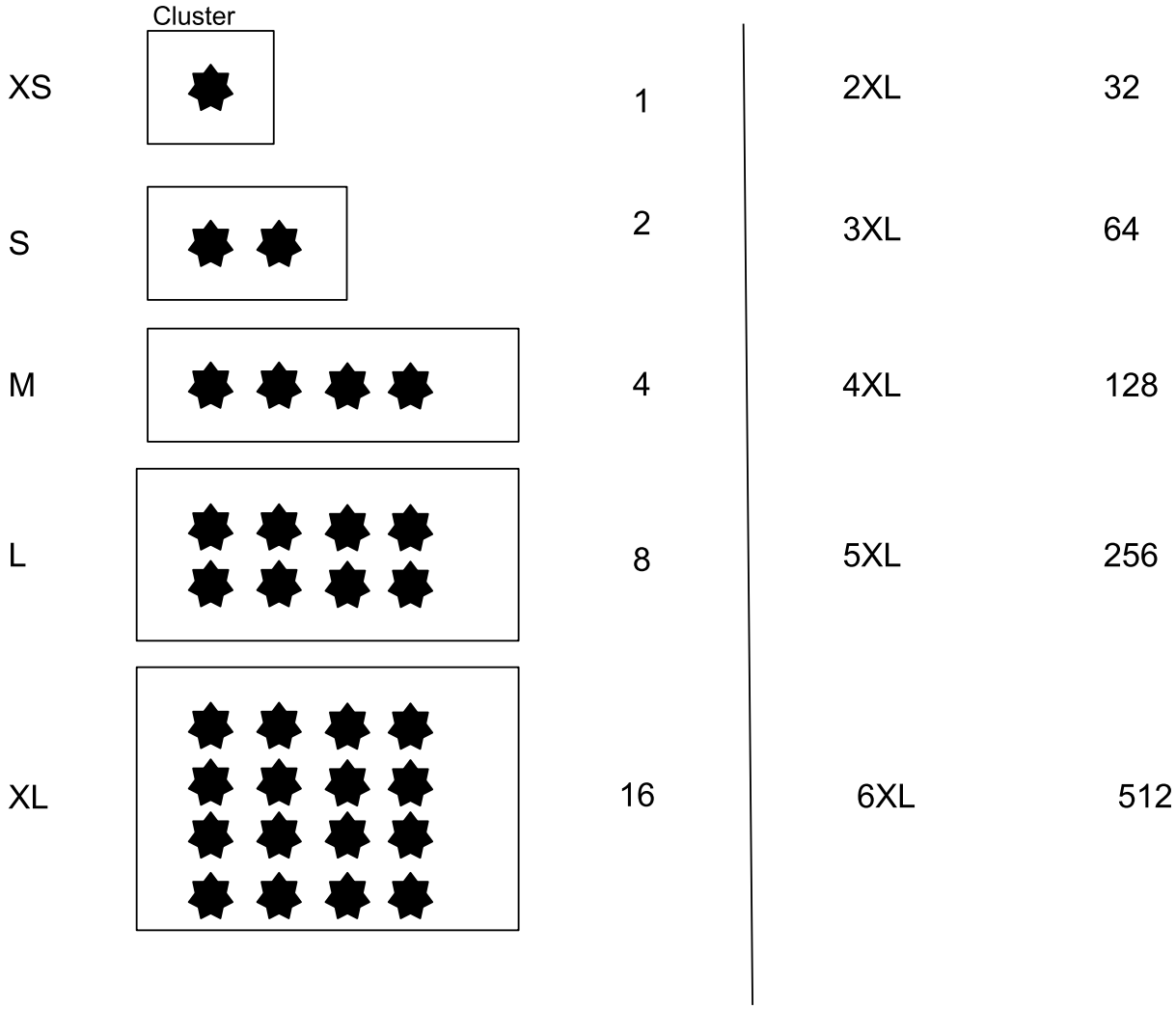
DATABASE STORAGE

Columnar storage
Stored as blobs in AWS S3 buckets
Not visible

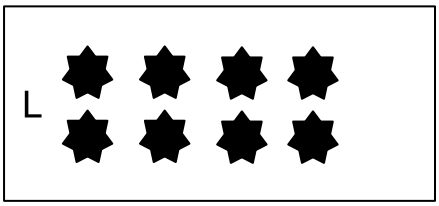
Three Layers:

1. **Database Storage**
    a. Data is optimized, compressed, **columnar format** ~ snowflake manages
    b. Optimized Data stored in cloud storage
    c. Not directly visible or accessible. Only by SQL in snowflake
2. **Query Processing**
    a. **Muscle of the system**
    b. Query execution using virtual warehouse
    c. MPP Compute Cluster
3. **Cloud Services**
    a. **Brain of the system**
    b. Coordinates activities such as - authentication, query parsing, metadata etc.

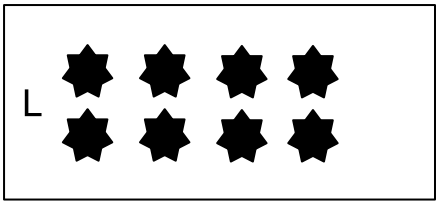
VIRTUAL WAREHOUSE SIZES



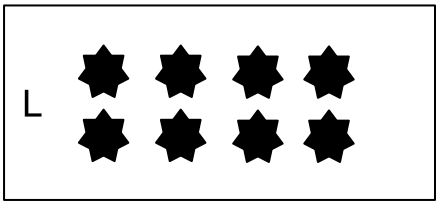
Single Cluster

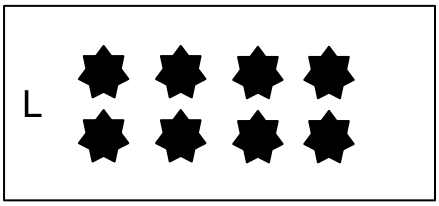


Multi-Clustering



concurrency..





Hands ON:

Since we talked about virtual warehouses, Let’s create it in Snowflake and understand its more details:

SCALING POLICY

Decides when to start additional cluster is started

Auto-Scale : Automatically adding or removing the clusters in multi-cluster warehouses.

Standard	Economy
Favors prevention of queuing	Favors conserving credits

Policy	Warehouse Starts..	Warehouse Shutdown..
Standard (default)	First warehouse **immediately** when query is queued. successive waits 20 seconds.	**2-3** checks every 1 min to determine if the least loaded warehouse can be distributed on another warehouse, without bringing least loaded back.
Economy	System estimates query load enough to keep new warehouse busy for **at least 6 minutes**	**5-6** checks every 1 min to determine if the least loaded warehouse can be distributed on another warehouse, without bringing least loaded back.

ALPHAEDGE SOLUTIONS

Hands on:

Create a database and table. Load the public available data into it to make your hands dirty.

```
//First DB Table and insert creation

create table ALPHAEDGE_FIRST_TABLE (
   FIRST_COLUMN    integer,
   SECOND_COLUMN   string,
   THIRD_COLUMN    string,
   FOURTH_COLUMN   string
   )
 ;

 insert into ALPHAEDGE_FIRST_TABLE values (1,'B','C','D' );

 select * from ALPHAEDGE_FIRST_TABLE ;



// Load a csv file publicly available in snowflake-docs tutorial. Reference snowflake documentation


// Create First DB
create or replace database ALPHAEDGE_FIRST_DB;

 // Set database
 USE DATABASE ALPHAEDGE_FIRST_DB;

create or replace table CONTACTS (
  id integer,
  last_name string,
  first_name string,
  company string,
  email string,
```

```
  workphone string,
  cellphone string,
  streetaddress string,
  city string,
  postalcode string);


//Validate data - there should not be any rows
 SELECT * FROM CONTACTS;


 //Load data from S3 bucket

 COPY INTO CONTACTS
    FROM s3://snowflake-docs/tutorials/dataloading/contacts1.csv
    file_format = (     type = 'CSV'
                                    field_delimiter = '|'
                                    skip_header = 1);


//Validate
 SELECT * FROM CONTACTS;
```

Snowflake Editions

STANDARD

Introductory

ENTERPRISE

For Large Scale Enterprise

VIRTUAL PRIVATE SNOWFLAKE

Highest level of security

BUSINESS CRITICAL

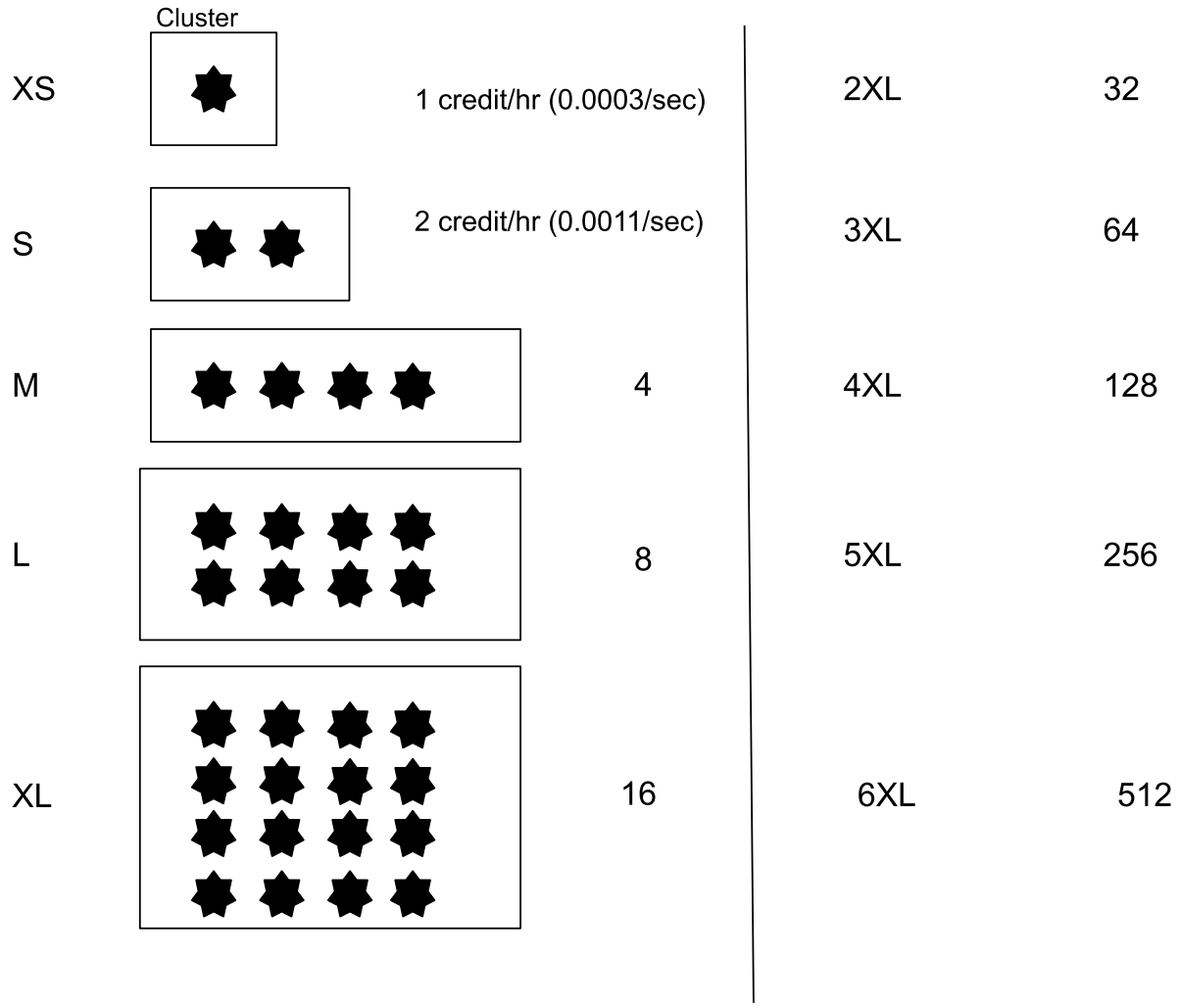
For organization with sensitive data which needs higher data protection

ALPHAEDGE SOLUTIONS

Standard	Enterprise	Business Critical	Virtual Private
SOC2 Type 2	All Standard	All Enterprise	All Business Critical
Federated Auth and SSO		Tri-secret Secure	Dedicated metadata store and compute resources
Oauth	Periodic Re-keying	Private connectivity Support using AWS/Azure private link	
Network Policies	Column Level Security		
Encryption of all data	Row Access Policy	PHI - HIPAA	No Data Sharing
Multi-factor Authentication	Object Tagging	PCI - DSS	
Access Control	Classify Sensitive Data	FedRAMP - US Gov region	
Time Travel upto 1 day	Time travel up to 90 days	IRAP - Asia pacific region	
DR for 7 days beyond time travel	ACCESS_HISTORY	Database Failover/Failback	
Virtual Warehouses	Multi-Cluster Virtual Warehouse	Redirect Client connection	
Resource Monitors	Search Optimization		
Database replication across accounts with in org	Materialized Views		

ALPHAEDGE SOLUTIONS

SNOWFLAKE PRICING ~ Understanding Credits



Credits for **user** managed virtual warehouses

- Size
- No (if multicluster).
- Time

Credits for s**nowflake** managed compute resources

- Service usage - 10% of compute usage free
- Serverless Features (snowflake managed compute resources)
    - Snowpipe
    - Database Replication
    - Materialized View Maintenance
    - Automatic Clustering
    - Search Optimization Service

Billed in Snowflake credits per seconds. (Minimum 1 minute)

Minimum 1-minute concept. Start/resume/resize charges one minute extra

Run Time	Charged Time
6 seconds	60 sec
65 seconds	65 sec
1sec, 30 sec	120 sec
60 seconds, 1 seconds, 30 seconds	180 sec (start/resume charges minimum 1 minute)
1 second, 20 second, 68 second	188 seconds

Small-Size single-cluster warehouse running for 2 hours

Time	Single Small Warehouse	Total Credit
Ist hour	2	2
Second hour	2	2
Total	4	4

Small-Size multi-cluster warehouse with 3 warehouses

1. Run Mode - Maximized (2 hrs)

Time	WH1	WH2	WH3	Total Credit
Ist Hr	2	2	2	6
Second Hr	2	2	2	6
Total Credit	4	4	4	12

ALPHAEDGE SOLUTIONS

2. Run Mode - Auto-scale (2 hrs)
    a. WH1 runs continuously
    b. WH2 runs continuously for the 2nd hr only
    c. WH3 runs for 30 minutes during 2nd hour

Time	WH1	WH2	WH3	Total Credit
Ist Hr	2	0	0	2
Second Hr	2	2	1	5
Total	4	2	1	7

3. Run Mode - Auto-scale (3 hrs) with Resize from
    a. WH1 runs continuously
    b. WH2 runs continuously for the 2nd hr only
    c. Multi-Cluster warehouse resized from small to medium at 1.30 hour
    d. WH3 runs for 15 min in the 3rd hour

Time	WH1	WH2	WH3	Total Credit
Ist Hr	2	0	0	2
Second Hr	1+2	1+2	0	6
Third Hr	4	4	1	9
Total Credit	9	7	1	17

ALPHAEDGE SOLUTIONS

Decoupled Compute and Storage pricing

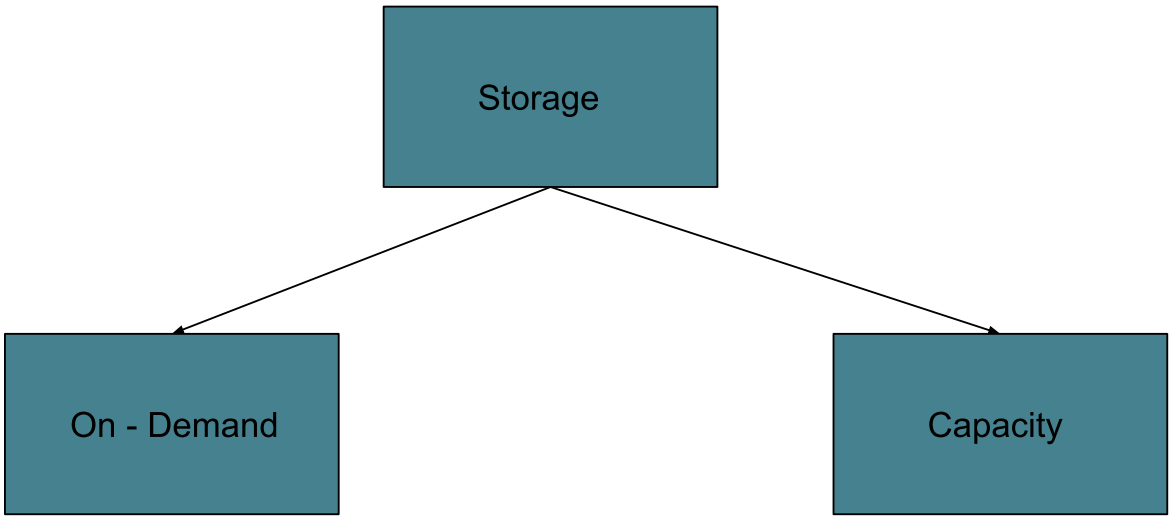
Compute	Storage
Billed in snowflake credits per seconds (min 1 minute)	Billed in dollars for average storage used per month
No of credits depends on size and no of warehouses active per hour	Columnar compressed storage is considered

ALPHAEDGE SOLUTIONS

Credits to Dollar and **Dollar for Storage** (not credit). It depends on:

1. Cloud Provider
2. Region
3. Edition

https://www.snowflake.com/pricing/



Pay for usage month to month

Pay for usage upfront

$40/TB

$23/TB

Use Case 1:
Actual used 0.25TB/Month
Bill: 0.25*$40= $10

Use Case 2:
Actual used 0.8TB/Month
Bill: 0.8*$40= $32

Use Case 1:
Actual used 0.25TB/Month
Bill: $23 (paid upfront)

Use Case 2:
Actual used 0.8TB/Month
Bill: $23 (paid upfront)

Best Practice - Start with On-Demand and later switch to Capacity, once more clarity on storage usage estimates.

<u>HandsOn to show Costs in Snowflake</u>

ALPHAEDGE SOLUTIONS

Performance Tuning in Snowflake

1. Why do we need performance tuning?

Faster execution

Cost Saving

ALPHAEDGE SOLUTIONS

2. What are traditional methods to improve performance in relational warehouses?

Table Designs

Primary Keys, Indexes

Table Partitioning

Check Query plans

Hints

ALPHAEDGE SOLUTIONS

3. What is the difference between traditional/static partitioning and snowflake partitioning?

Traditional Partitions	Micro-Partitions
Created using DDL	Automatically performed
Maintenance Overhead	No limitations, Additional benefits
Size depends on partition key	50MB to 500MB, ..16MB compressed
	IMMUTABLE

Snowflake automatically leverages the micro-partitions to improve performance.

ALPHAEDGE SOLUTIONS

4. What can Snowflake System admins or developers do to improve performance?

    a. Choose the right size of virtual warehouse.
    b. Should split files into smaller sizes.
    c. Create right cluster keys on big tables.
    d. Understand the storage requirements - Capacity Vs on-demand

User group / Workload specific virtual warehouse - (Dedicated)

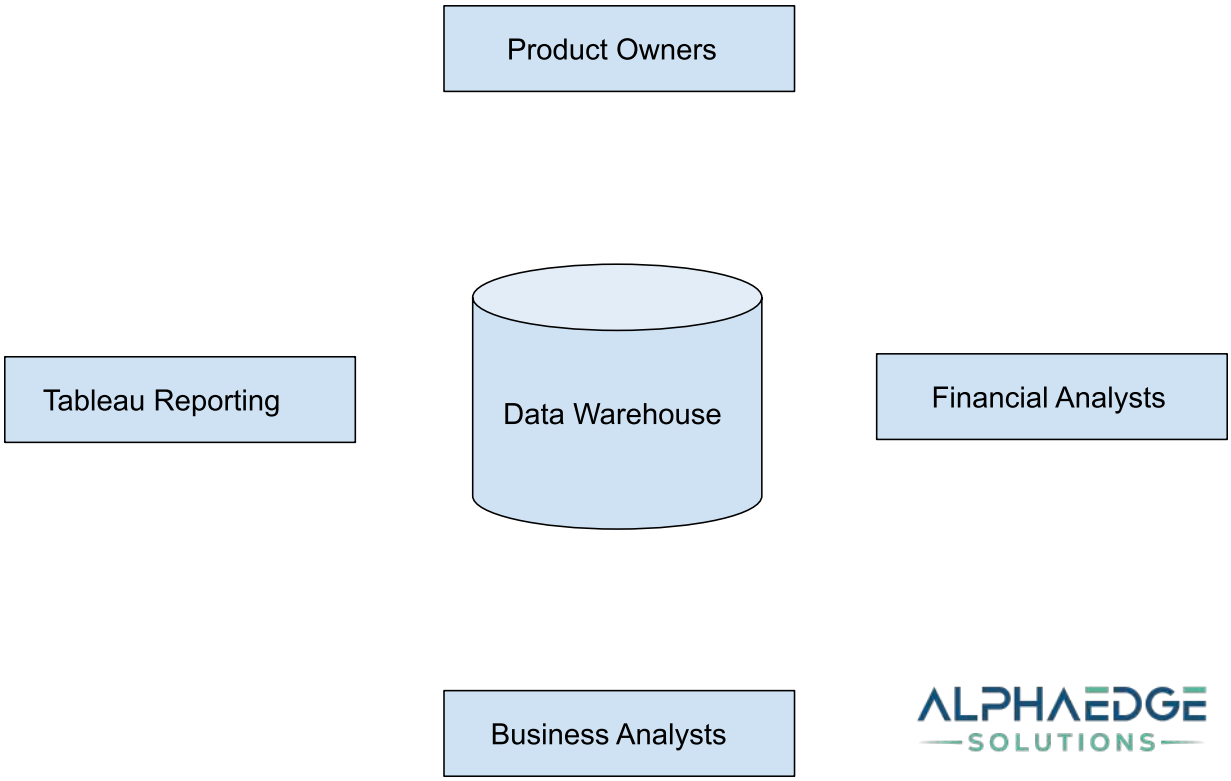
Scaling up - If period of complicated workload is known

Scaling out - If concurrency is not known

Automatic Caching - Maximize automatic caching

Clustering - For table with size in terabytes

Dedicated Virtual Warehouses



1. Understand the system to find out user groups and workloads
2. Create the dedicated virtual warehouses for user groups OR Workloads
3. Keep an eye on utilization of dedicated virtual warehouses. (Neither too much or too less)
4. Increase or decrease dedicated virtual warehouses based on system usage.

Hands on - Dedicated Virtual Warehouses

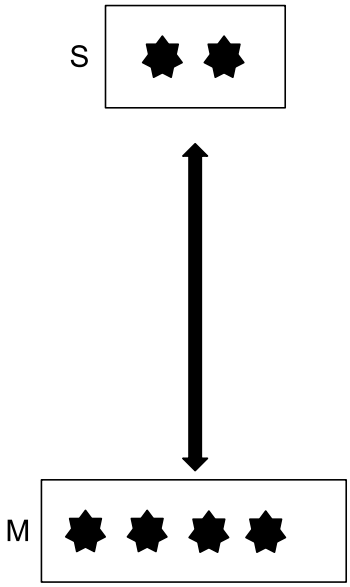
```
// Dedicated Warehouse for Product Owners and Financial Analysts

// Product Owners

CREATE WAREHOUSE PRODUCT_OWNERS_WH
WITH WAREHOUSE_SIZE = 'SMALL'
WAREHOUSE_TYPE = 'STANDARD'
AUTO_SUSPEND = 600
AUTO_RESUME = TRUE
MIN_CLUSTER_COUNT = 1
MAX_CLUSTER_COUNT = 2
SCALING_POLICY = 'ECONOMY';

// Financial analysts

CREATE WAREHOUSE FINANCIAL_ANALYSTS_WH
WITH WAREHOUSE_SIZE = 'XSMALL'
WAREHOUSE_TYPE = 'STANDARD'
AUTO_SUSPEND = 600
AUTO_RESUME = TRUE
MIN_CLUSTER_COUNT = 1
MAX_CLUSTER_COUNT = 2
SCALING_POLICY = 'ECONOMY';
```

// Create roles for above user groups. Use Account Admin to handle roles

```
CREATE ROLE PRODUCT_OWNERS_ROLE;
GRANT USAGE ON WAREHOUSE PRODUCT_OWNERS_WH TO ROLE PRODUCT_OWNERS_ROLE;

CREATE ROLE FINANCIAL_ANALYSTS_ROLE;
GRANT USAGE ON WAREHOUSE FINANCIAL_ANALYSTS_WH TO ROLE FINANCIAL_ANALYSTS_ROLE;


// Create some users

// Product Owners  - Adam, Smith, John, Alex

CREATE USER ADAM PASSWORD = 'ADAM'
LOGIN_NAME = 'ADAM'
DEFAULT_ROLE='PRODUCT_OWNERS_ROLE'
DEFAULT_WAREHOUSE = 'PRODUCT_OWNERS_WH'
MUST_CHANGE_PASSWORD = FALSE;

CREATE USER  SMITH PASSWORD = 'SMITH'
LOGIN_NAME = 'SMITH'
DEFAULT_ROLE='PRODUCT_OWNERS_ROLE'
DEFAULT_WAREHOUSE = 'PRODUCT_OWNERS_WH'
MUST_CHANGE_PASSWORD = FALSE;

CREATE USER  JOHN PASSWORD = 'JOHN'
LOGIN_NAME = 'JOHN'
DEFAULT_ROLE='PRODUCT_OWNERS_ROLE'
DEFAULT_WAREHOUSE = 'PRODUCT_OWNERS_WH'
MUST_CHANGE_PASSWORD = FALSE;

CREATE USER  ALEX PASSWORD = 'ALEX'
LOGIN_NAME = 'ALEX'
DEFAULT_ROLE='PRODUCT_OWNERS_ROLE'
DEFAULT_WAREHOUSE = 'PRODUCT_OWNERS_WH'
MUST_CHANGE_PASSWORD = FALSE;

GRANT ROLE PRODUCT_OWNERS_ROLE TO USER ADAM;
GRANT ROLE PRODUCT_OWNERS_ROLE TO USER SMITH;
GRANT ROLE PRODUCT_OWNERS_ROLE TO USER JOHN;
GRANT ROLE PRODUCT_OWNERS_ROLE TO USER ALEX;


SHOW USERS;

DESC USER ADAM;

//Reset Password generates a URL to share with User.

//Existing password remain valid until changed by user

// Generated URL expires in 4 hrs

// Disabling the user
   //Terminates existing session
   // User immediately locked out
```

```
ALTER USER ADAM RESET PASSWORD ;


// Financial Analysts  - Ashley, Bob, Jessica

CREATE USER ASHLEY PASSWORD = 'ASHLEY'
LOGIN_NAME = 'ASHLEY'
DEFAULT_ROLE='FINANCIAL_ANALYSTS_ROLE'
DEFAULT_WAREHOUSE = 'FINACIAL_ANALYSTS_WH'
MUST_CHANGE_PASSWORD = FALSE;

CREATE USER BOB PASSWORD = 'BOB'
LOGIN_NAME = 'BOB'
DEFAULT_ROLE='FINANCIAL_ANALYSTS_ROLE'
DEFAULT_WAREHOUSE = 'FINACIAL_ANALYSTS_WH'
MUST_CHANGE_PASSWORD = FALSE;

CREATE USER JESSICA PASSWORD = 'JESSICA'
LOGIN_NAME = 'JESSICA'
DEFAULT_ROLE='FINANCIAL_ANALYSTS_ROLE'
DEFAULT_WAREHOUSE = 'FINACIAL_ANALYSTS_WH'
MUST_CHANGE_PASSWORD = FALSE;



GRANT ROLE FINANCIAL_ANALYSTS_ROLE TO USER ASHLEY;
GRANT ROLE FINANCIAL_ANALYSTS_ROLE TO USER BOB;
GRANT ROLE FINANCIAL_ANALYSTS_ROLE TO USER JESSICA;



// Clean up

DROP USER ADAM;
DROP USER SMITH;
DROP USER JOHN;
DROP USER ALEX;

DROP USER ASHLEY;
DROP USER BOB;
DROP USER JESSICA;


DROP ROLE PRODUCT_OWNERS_ROLE;
DROP ROLE FINANCIAL_ANALYSTS_ROLE;

DROP WAREHOUSE PRODUCT_OWNERS_WH;
DROP WAREHOUSE FINANCIAL_ANALYSTS_WH;
```

Scale Up and Scale Down



For known workloads

Example : Batch run - Morning - Start Day Batch/ Evening (End of Day batch)

Solves running complex queries faster

This is feature is not for high concurrency issues or more users.

For more users - Scale out via Automatic Multi-Clustering is best

Sizing considerations

Use Auto-suspend, bigger size are costly

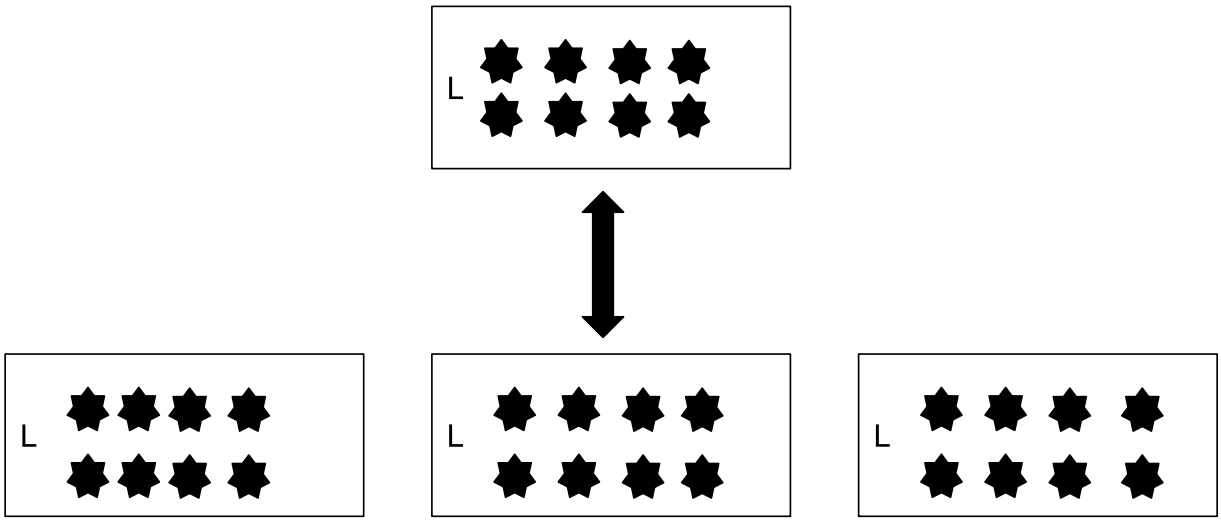
Identify similar size / complexity workload and allocate the warehouse (dedicated)

Scale Up Hands On

SQL

```
--See the warehouses
SHOW WAREHOUSES ;

--Scale Up
ALTER WAREHOUSE ALPHAEDGE_TESTWH SET WAREHOUSE_SIZE = 'MEDIUM';

--See the warehouses
SHOW WAREHOUSES ;

--Scale Down
ALTER WAREHOUSE ALPHAEDGE_TESTWH SET WAREHOUSE_SIZE = 'XSMALL';
--See the warehouses
SHOW WAREHOUSES ;
```

Scaling Out for concurrency



Single WH, Multiple Cluster

Handles large number of concurrent users

Automatic process of adding and removing clusters based on usage

Clusters deployed across AZ's - for high availability

Queries are load balanced across clusters in virtual warehouses.

Things to Note
Available in Business Enterprise and above
MIN_CLUSTER_COUNT: 1-10 (default 1)
MAX_CLUSTER_COUNT: 1-10 (default 1) - can be high to solve concurrency issues.

Differences - since sounds same (can be questions in exam if writing snowpro)

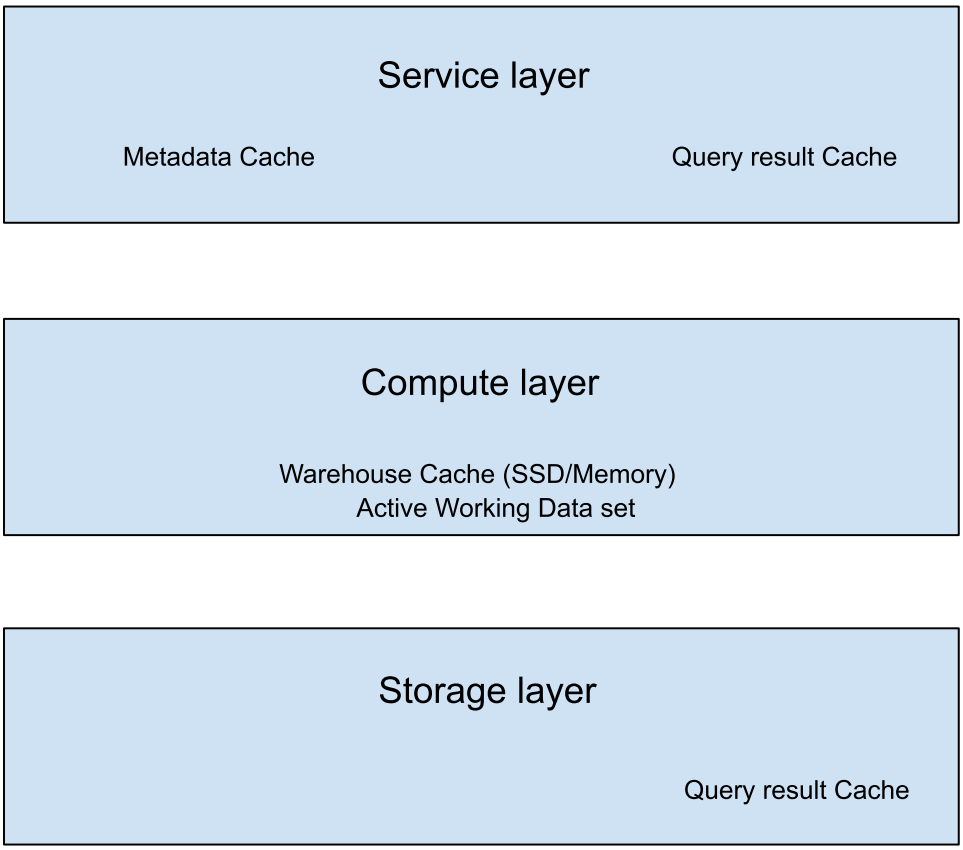
Scaling Up	**Scaling Out**
Increase Size of Virtual Warehouse To Solve Complex Query issues	Addition of Clusters in Multi-Cluster Warehouse To solve more users / queries

Scale Out / In Hands on

SHOW WAREHOUSES ;

SELECT * FROM catalog_sales ;

Caching



Summary	Metadata Cache	Data Cache	Query Result Cache
Location	Service Layer	Warehouse SSD	Service Layer / Storage layer
Contents	Metadata and stats for tables and MP (micro-partitions)	Active Data set, not the results	Result sets
Utilization	No compute is used. Aggregation - Min, max, count Show commands Information schema queries	When a query is run which uses some or all the active data set	When the same query is run. Underlying data is not changed
Age	No age, updated continuously	While the warehouse is active. It uses the **least recently used** algorithm to removed content from cache	24 hours, This 24 hrs resets every time a query is run.

1. Metadata Cache
    a. Stores metadata about tables and micro partitions

    b. Snowflake does not use compute for range e.g MIN, MAX, counts - row and nul

2. Local disk Cached or Virtual Warehouse Cache
    a. Cache data sets used by SQL queries
    b. Retrieved from Remote Disk, stored in SSD / Memory
    c. Clears when Virtual warehouse is suspended
        i. alter warehouse compute_wh suspend ;
3. Result Cached
    a. Results of every query executed in the last 24 hrs. Same queries reuse results.
    b. Available across virtual warehouses. If data is unchanged other users can use it.
    c. Can be disabled at session level. Mainly for benchmarking purpose
        i. alter session set use_cached_result = false ;

Guidelines to Reuse Cache

1. Same queries should go to the same warehouse for result cache
2. User groups (Different users running same queries)
3. Similar queries using the same underlying data set to use the warehouse cache. (underlying data not changed)

Maximize Caching HandsOn

```
--Check that the warehouse in suspended state. The warehouse cashe is cleared if it is in suspended state.
 show warehouses ;

-- Running First Time - Warehouse cache will not be used and result cache will not be used
alter session set use_cached_result = false ;

-- Lets run a little complicated query which comes back in few seconds
select ca_state, ca_city, count(*)
from customer_address
join customer on c_current_addr_sk = ca_address_sk
where ca_state in ('NC','OH','NY','TX','MT','VA','OK','GA','MN','KY')
group by ca_state, ca_city;

-- Check the query profile, warehouse cache should not have been used


-- Run the query again
select ca_state, ca_city, count(*)
from customer_address
join customer on c_current_addr_sk = ca_address_sk
where ca_state in ('NC','OH','NY','TX','MT','VA','OK','GA','MN','KY')
group by ca_state, ca_city;

-- check the query profile. This time Warehouse Cache must have been used

-- Now lets enable the result cache
alter session set use_cached_result = true ;
```

```
-- Run the query again
select ca_state, ca_city, count(*)
from customer_address
join customer on c_current_addr_sk = ca_address_sk
where ca_state in ('NC','OH','NY','TX','MT','VA','OK','GA','MN','KY')
group by ca_state, ca_city;

-- Check the query Profile it should use result cache

-- Now lets see if a different user sends the same query to the warehouse. It should use result cache
-- Note result cache is true by default, we have set it false to show the usage of warehouse cache

-- create role
CREATE ROLE PRODUCT_OWNERS_ROLE;

-- Product owner role

GRANT USAGE ON WAREHOUSE COMPUTE_WH TO ROLE PRODUCT_OWNERS_ROLE;

-- create user
CREATE USER ADAM PASSWORD = 'ADAM'
LOGIN_NAME = 'ADAM'
DEFAULT_ROLE='PRODUCT_OWNERS_ROLE'
DEFAULT_WAREHOUSE = 'COMPUTE_WH'
MUST_CHANGE_PASSWORD = FALSE;

--grant role
GRANT ROLE PRODUCT_OWNERS_ROLE to USER ADAM;


-- Cleanup

-- drop role
drop role PRODUCT_OWNERS_ROLE ;

-- drop user
DROP USER ADAM ;

-- suspend warehouse
 alter warehouse compute_wh resume;

 alter warehouse compute_wh suspend;

 -- show warehouses
 show warehouses ;
```

Clustering

What is a Clustering Key?

Clustering Key - subset of columns (or expressions) designed to co-locate similar rows in same micro-partitions.

Hands On.

```
-- Table with out cluster Key by columns
create or replace table t0 (c1 date, c2 string) ;

show tables like 't0';

--Clustering by columns
create or replace table t1 (c1 date, c2 string, c3 number) cluster by (c1, c2);
```

```
show tables like 't1';

-- clustering by expressions
create or replace table t2 (c1 timestamp, c2 string, c3 number) cluster by (to_date(c1), substring(c2, 0, 10));

show tables like 't2';

-- cluster by paths in variant columns
create or replace table t3 (t timestamp, v variant) cluster by (v:"Data":id::number);

show tables like 't3';


-- When you try to get the clustering info on non clustered table. It gives error
select system$clustering_information('T0') ;

-- Clustering information based on column
select system$clustering_information('T2','(c1)') ;


--- Let see some real tables..change DB and schema (sample, 100TCL)
select system$clustering_information('catalog_sales') ;

-- can see depth independently
select system$clustering_depth ('catalog_sales') ;

-- customer table
select system$clustering_information('customer') ;

-- can see depth independently
select system$clustering_depth ('customer') ;


-- Lets see the table we created. Now can we add the cluster Key

-- Gives error since no cluster key,
select system$clustering_depth ('contacts') ;

-- alter to add the cluster key
alter table contacts cluster by (city) ;

show tables like 'contacts'

-- More clustering information, automatic clustering off
select system$clustering_information('contacts') ;

-- Automatic Clustering
```

```
-- Clustering are ON by default, the moment you define a clustering key

-- The tables created by (create table..clone..) from a source table that has a clustering key.
-- The cloned table will have the automatic clustering suspended.
-- Lets see

create table clone_contacts clone contacts ;

show tables like 'clone_contacts'

-- Resume automatic {manual reclustering deprecated}
alter table clone_contacts resume recluster ;

show tables like 'clone_contacts'

---- Suspend automatic
alter table clone_contacts suspend recluster ;

show tables like 'clone_contacts'

-- Drop clustering

alter table clone_contacts drop clustering key ;
show tables like 'clone_contacts'

alter table contacts drop clustering key ;
show tables like 'contacts'

-- Clustering cost for a specified table

select * from table(automatic_clustering_history(table_name=>'information_schema.catalog_sales')) ;

select *
  from table(information_schema.automatic_clustering_history(
    date_range_start=>'2022-05-10 13:00:00.000 -0700',
    date_range_end=>'2022-05-16 14:00:00.000 -0700'));


--- recluster
-- alter to add the cluster key
alter table contacts cluster by (city) ;

show tables like 'contacts'

-- recluster
alter table contacts cluster by (city,postalcode) ;

show tables like 'contacts'
```

```
-- It will only show reclustering if it is done in that time period. since the table is small no reclustering is done.
select *
  from table(information_schema.automatic_clustering_history(
    date_range_start=>'2022-05-10 13:00:00.000 -0700',
    date_range_end=>'2022-05-16 14:00:00.000 -0700'));
```

HandsOn to see performance improvement. alphaedge_first_db.public

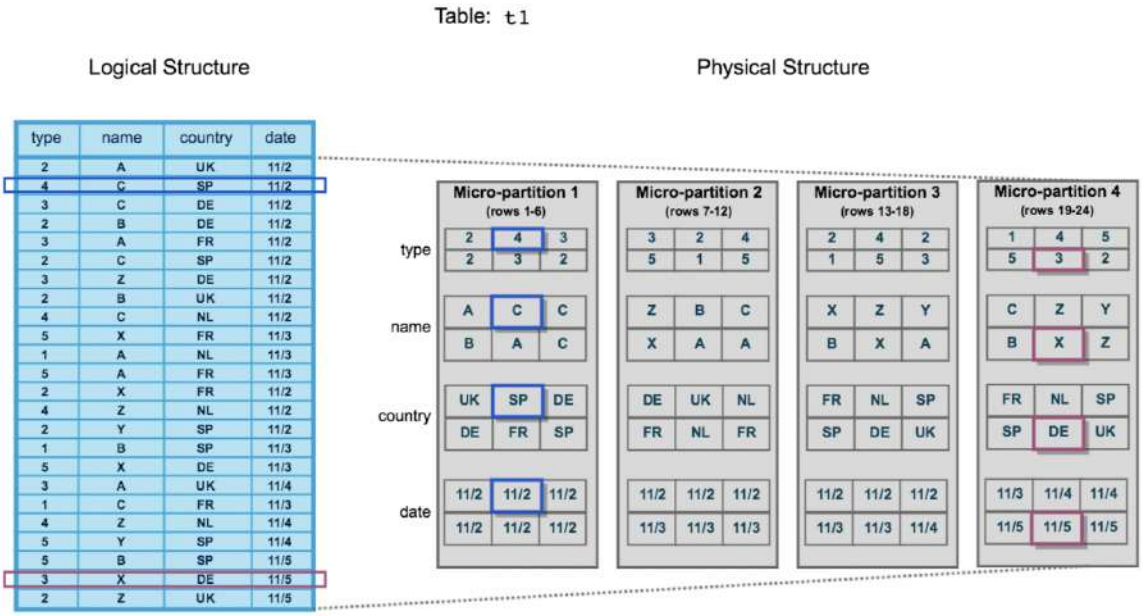
```
--- Performance improvement hands on via clustering
drop table if exists customer ;

create table customer as select * from snowflake_sample_data.tpcds_sf100tcl.customer ;
select system$clustering_information('customer') ; -- error hence there is no clustering on the table

select count(*) from customer where c_birth_year in (1962) ;
--Add cluster key
alter table customer cluster by (c_birth_year);
select system$clustering_information('customer') ;

select count(*) from customer where c_birth_year in (1965) ;
alter table customer drop clustering key ;
```

What are Micro Partitions?

MPs (micro-partition) : Table data is stored in files of size 50-500MB. These are contiguous storage units.

The logical structure of the table is stored into physical structure in micro partitions as per the clustering Key. Snowflake maintained.

Micro Partitions are small files of size 50 - 500MB.



What happens when a clustering key is altered?

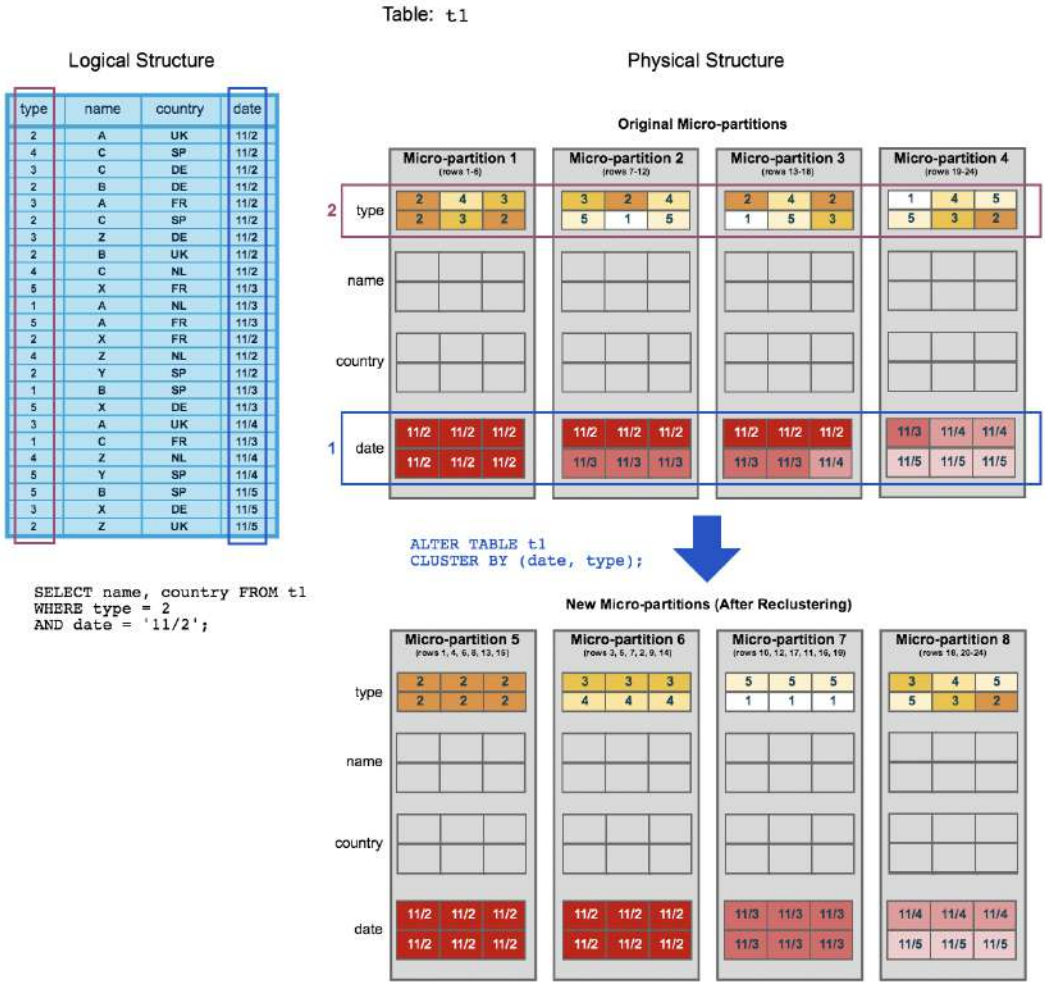
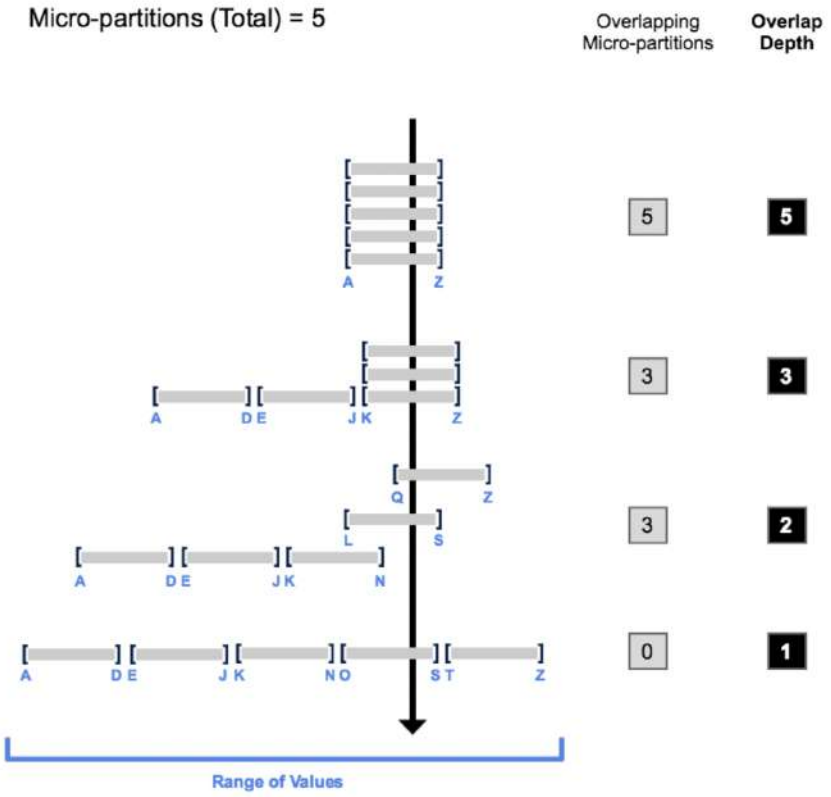


Diagram Ref: Snowflake documentation

Immutability - How Insert, Update, Delete works.

Clustering information - MP’s Or Query Pruning and Depths

What is Micro-partition Overlap and depth?



Why is Clustering needed?

It increases scan efficiency for large tables.

Generally snowflake produces well clustered data in tables

Over a period of time, tables might not be clustered optimally. Hence Reclustering is done.

So re-clustering is done manually.

How to check if clustering is needed?

If performance degrades for large (multi TB) tables. Clustering is an option but comes with a cost.

Which columns should be used as Cluster Keys?

Snowflake recommends maximum 3-4 columns in a cluster key

Columns which are used in join and where clauses

Columns of high cardinality and low cardinality both should be avoided (Date and Sex)

What are Clustering Guidelines and Best Practices?

1. Do not cluster if there is no performance issue.
2. Clustering sorts and bucket the data, hence choose keys wisely
3. Cluster only large tables ~ bigger than 500GB size
4. Smaller table- try clustering only if you see real performance issues.
5. Not too many columns in the clustering key. 3 -4 is good.
6. Function based clause in joins and where should be avoided, unless clustering is done on it.

Alternative to clustering

1. Insert the sorted data on keys
2. This will reduce clustering cost.

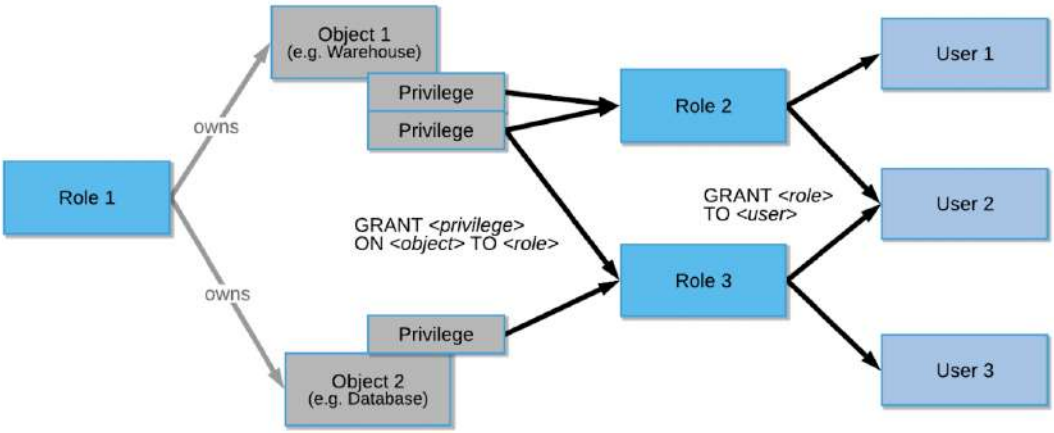
Access Control

DAC	RBAC
Discretionary access control	Role based access control
Object has owner Owner can only grant access	Access Privs assigned to roles Roles to users

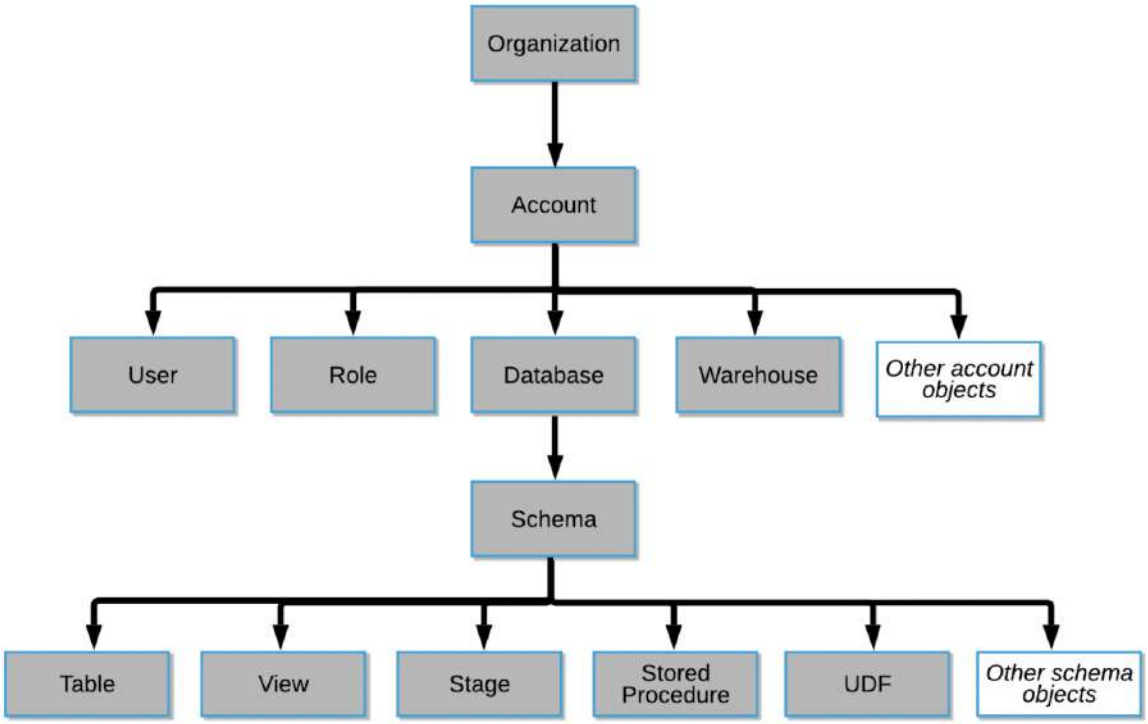
Key Concepts

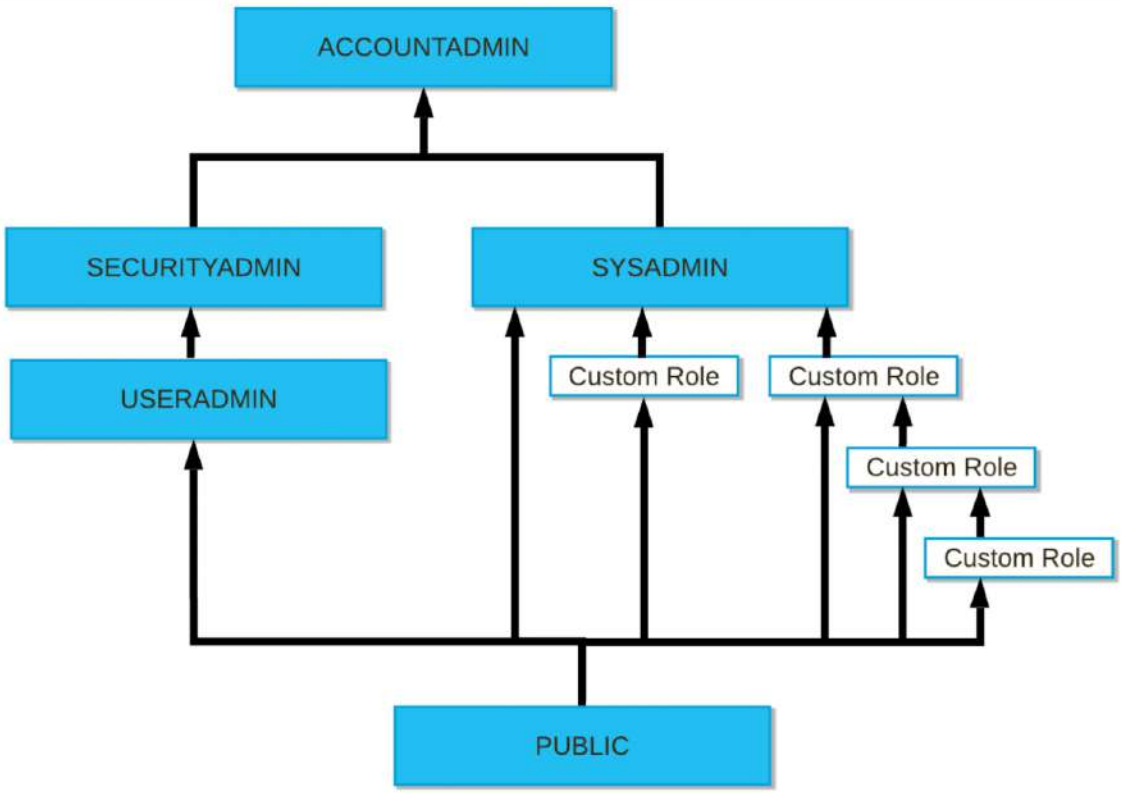
Securable Object	Privilege	Role	User
Entity to which access can be granted (warehouse, tables, databases, schemas ect)	Defined level of access	System Defined	User identity recognized by snowflake
Access strictly controlled by grant	Select, usage, ownership, delete Usage etc.	User Defined	Can be a person or program

RBAC



Securable Objects





ORGADMIN	ACCOUNTADMIN	SECURITYADMIN	USERADMIN	SYSADMIN	PUBLIC
Create accounts	Top level role (account administrator)	Manage grants globally	User Management only	Create warehouse and databases	Auto-granted role.
View all accounts	Should be granted to controlled users	Manage users and roles	Role Management only	Custom Roles assign to this role	Full access in account
View regions	Has all privs of securityadmin and sysadmin	Has all privs of useradmin			

Tables in Snowflake

Permanent	Transient	Temporary
create table	create **transient** table	create **temporary** table
not session specific	not session specific	session specific
time travel retention-> 0 - 90 days	time travel retention-> 0 - 1 day	time travel retention-> 0 - 1 day
fail safe	no fail safe	no fail safe
use for high data protection	use for no data protection	use for temporary data processing

Why is it important to understand the table types ?

To manage storage cost efficiently. You will use a specific table type as per the requirement.

Which table is visible if all three types of table have the same name? If a table named “client” is present as permanent, transient and temporary and I execute “select * from client”, which table will return the data? Is there any conflict in names?

Lets see in the demo.

```
-- drop table if exists client ;
-- drop table if exists person ;

show tables ; -- kind => TABLE for permanent,  kind = TRANSIENT,  kind = TEMPORARY

create table client
(  id NUMBER(38,0), first_name VARCHAR(16), last_name VARCHAR(50),
   sex VARCHAR(1), ethinicity VARCHAR(30), ssn VARCHAR(15),
   street_address VARCHAR(90),status VARCHAR(10)
```

```
);

insert into client values (111111, 'James',     'Schwartz', 'M', 'American','342-76-9087','5676 Washington
Street','ACTIVE') ;
insert into client values (222222, 'Jessica',   'Escobar',  'F', 'Hispanic','456-93-5629','3234 WateringCan
Drive','INACTIVE') ;
insert into client values (333333, 'Ben',       'Hardy',    'M', 'American','876-98-3245','6578 Historic
Circle','INACTIVE') ;
insert into client values (444444, 'Anjali',    'Singh',    'F', 'Indian American','435-87-6532','8978 Autumn
Day Drive','ACTIVE') ;
insert into client values (555555, 'Dean',      'Tracy',    'M', 'African','767-34-7656','2343 India
Street','ACTIVE') ;

select * from client ;

create transient table client
(  id NUMBER(38,0), first_name VARCHAR(16), last_name VARCHAR(50),
   sex VARCHAR(1), ethinicity VARCHAR(30), ssn VARCHAR(15),
   street_address VARCHAR(90),status VARCHAR(10)
); -- can't create


-- but can create the temporary table of the same name
-- and which will hide the main permanent table client thats what we will see..
create temporary table client
(  id NUMBER(38,0), first_name VARCHAR(16), last_name VARCHAR(50),
   sex VARCHAR(1), ethinicity VARCHAR(30), ssn VARCHAR(15),
   street_address VARCHAR(90),status VARCHAR(10)
);

insert into client values (444444, 'Anjali',    'Singh',    'F', 'Indian American','435-87-6532','8978 Autumn
Day Drive','ACTIVE') ;
insert into client values (555555, 'Dean',      'Tracy',    'M', 'African','767-34-7656','2343 India
Street','ACTIVE') ;

-- Hence any select or DML operation will go to a temporary table.
select * from client ;

show tables ;

-- if you need to go back to the permanent table. you need to close the session
-- OR drop the table, it will drop the temporary table.

drop table client ;

select * from client ; -- Now it goes back to the permanent table.
```

```
-------------------------------
-- transient table
create transient table person
(  id NUMBER(38,0), first_name VARCHAR(16), last_name VARCHAR(50),
   sex VARCHAR(1), ethinicity VARCHAR(30), ssn VARCHAR(15),
   street_address VARCHAR(90),status VARCHAR(10)
);

show tables ;



insert into person values (111111, 'James',     'Schwartz', 'M', 'American','342-76-9087','5676
Washington Street','ACTIVE') ;
insert into person values (222222, 'Jessica',   'Escobar',  'F', 'Hispanic','456-93-5629','3234
WateringCan Drive','INACTIVE') ;
insert into person values (333333, 'Ben',       'Hardy',    'M', 'American','876-98-3245','6578 Historic
Circle','INACTIVE') ;
insert into person values (444444, 'Anjali',     'Singh',   'F', 'Indian American','435-87-6532','8978 Autumn
Day Drive','ACTIVE') ;
insert into person values (555555, 'Dean',      'Tracy',    'M', 'African','767-34-7656','2343 India
Street','ACTIVE') ;

select * from person ;


create  table person
(  id NUMBER(38,0), first_name VARCHAR(16), last_name VARCHAR(50),
   sex VARCHAR(1), ethinicity VARCHAR(30), ssn VARCHAR(15),
   street_address VARCHAR(90),status VARCHAR(10)
);


-- temporary table hides the main transient table person
create temporary table person
(  id NUMBER(38,0), first_name VARCHAR(16), last_name VARCHAR(50),
   sex VARCHAR(1), ethinicity VARCHAR(30), ssn VARCHAR(15),
   street_address VARCHAR(90),status VARCHAR(10)
);

insert into person values (333333, 'Ben',       'Hardy',    'M', 'American','876-98-3245','6578 Historic
Circle','INACTIVE') ;
insert into person values (444444, 'Anjali',     'Singh',   'F', 'Indian American','435-87-6532','8978 Autumn
Day Drive','ACTIVE') ;
insert into person values (555555, 'Dean',      'Tracy',    'M', 'African','767-34-7656','2343 India
Street','ACTIVE') ;

-- Hence any select or DML operation will go to a temporary table. Here we will open a new session
select * from person ;
```

-- Lets create a new worksheet, that will initiate a new session and will do a select on person

-- it shows 3 records in new session and 5 rows in previous session

-- new worksheet

select * from person ;

Database Types

Permanent	Transient
create database	create **transient** database
fail safe	no fail safe
Schema can be permanent/transient and table created can be permanent/transient/temporary	all schemas and underlying tables created are transient

Demo:

-- create transient database

create or replace transient database training_trnsnt ;
use database training_trnsnt ;

```
create or replace schema demo_prmnt ;

create or replace transient schema demo_trnsnt ;

show schemas ;
```

Schema Types

Permanent	Transient
create schema	create **transient** schema
fail safe	no fail safe
table created can be permanent/transient/temporary	all tables created are transient

```
use schema demo_trnsnt ;

create table client  -- tried creating a permanent table
(  id NUMBER(38,0), first_name VARCHAR(16), last_name VARCHAR(50),
   sex VARCHAR(1), ethinicity VARCHAR(30), ssn VARCHAR(15),
   street_address VARCHAR(90),status VARCHAR(10)
);

show tables ; -- kind shows transient
```

```
-- try creating temporary table

create temporary table client
(  id NUMBER(38,0), first_name VARCHAR(16), last_name VARCHAR(50),
   sex VARCHAR(1), ethinicity VARCHAR(30), ssn VARCHAR(15),
   street_address VARCHAR(90),status VARCHAR(10)
);

show tables ; -- kind shows temporary

-- Hence you can not create a permanent table in a transient schema. Only transient and temporary
tables are allowed.
```

USER DEFINED FUNCTIONS

1. Developers/Users create these functions to achieve various results
2. Can be scalar or tabular
    a. Scalar - returns the value of an expression or a single value
    b. Tabular - returns a table or a set of rows
3. Can be defined in SQL, java, javascript and python
4. Can be defined as secure or unsecure
5. Demo of scalar and tabular UDF in SQL

```
drop function area_of_triangle (float, float);

-- Scalar
create function fn_area_of_triangle (base float, height float)
   returns float
   as
   $$
      0.5*base*height
   $$
;

select fn_area_of_triangle (10,20) ;
```

```
-- Tabular user defined function

show tables  ;

drop table client ;

create table client
(  id NUMBER(38,0), first_name VARCHAR(16), last_name VARCHAR(50),
   sex VARCHAR(1), ethinicity VARCHAR(30), ssn VARCHAR(15),
   street_address VARCHAR(90),status VARCHAR(10)
);

insert into client values (111111, 'James',     'Schwartz', 'M', 'American','342-76-9087','5676 Washington Street','ACTIVE') ;
insert into client values (222222, 'Jessica',   'Escobar',  'F', 'Hispanic','456-93-5629','3234 WateringCan Drive','INACTIVE') ;
insert into client values (333333, 'Ben',       'Hardy',    'M', 'American','876-98-3245','6578 Historic Circle','INACTIVE') ;
```

```
insert into client values (444444, 'Anjali',     'Singh',   'F', 'Indian American','435-87-6532','8978 Autumn
Day Drive','ACTIVE') ;
insert into client values (555555, 'Dean',      'Tracy',    'M', 'African','767-34-7656','2343 India
Street','ACTIVE') ;

select * from client ;

-- tabular function

drop function fn_acctive_client  (varchar) ;

create or replace function fn_active_client (p_status varchar)
   returns table (id number, first_name varchar, last_name varchar, sex varchar, ssn varchar, status
varchar)
   as
   $$
      select id, first_name, last_name, sex, ssn, status from client where status = p_status
   $$
;


select id, first_name, last_name, sex, ssn, status from table(fn_active_client ('ACTIVE')) ;
```

STORED PROCEDURES

1. Theory
    a. Allows you to write procedural code to execute the SQL or operations in DB
    b. Called as independent statements
    c. Returning a value is optional.

2. Example: Stored procedures using snowflake scripting

Demo:

```
create or replace table client_stg
(  id NUMBER(38,0), first_name VARCHAR(16), last_name VARCHAR(50),
   sex VARCHAR(1), ethinicity VARCHAR(30), ssn VARCHAR(15),
   street_address VARCHAR(90),status VARCHAR(10)
);


delete from client_stg ;
insert into client_stg values (111111, 'James',     'Schwartz', 'M', 'American','342-76-9087','5676 Washington Street','ACTIVE') ;
insert into client_stg values (222222, 'Jessica',   'Escobar',  'F', 'Hispanic','456-93-5629','3234 WateringCan Drive','INACTIVE') ;
insert into client_stg values (333333, 'Ben',       'Hardy',    'M', 'American','876-98-3245','6578 Historic Circle','INACTIVE') ;
insert into client_stg values (444444, 'Anjali',    'Singh',    'F', 'Indian American','435-87-6532','8978 Autumn Day Drive','ACTIVE') ;
insert into client_stg values (555555, 'Dean',      'Tracy',    'M', 'African','767-34-7656','2343 India Street','ACTIVE') ;
```

```
select * from client_stg ;

-- create a target table

create or replace table client_tgt
(  id NUMBER(38,0), first_name VARCHAR(16), last_name VARCHAR(50),
   sex VARCHAR(1), ethinicity VARCHAR(30), ssn VARCHAR(15),
   street_address VARCHAR(90),status VARCHAR(10)
);


-- create a procedure
  create or replace procedure prc_load_client_tgt()
  returns varchar
  language sql
  as
  $$
    begin
      insert into client_tgt
         select * from client_stg ;

      return 'Record Inserted' ;

   end;
  $$
  ;

-- test the procedure
select * from client_stg ;
select * from client_tgt ; -- empty
call prc_load_client_tgt() ;
select * from client_tgt ; -- procedure works
```

UDF and SP comparison

Characteristic	UDF	SP
Return	Always return a value	Can return a value
Call	Called in SQL	Called independently
Usage of output	Since called in SQL hence its value can be used in SQL	The value can not be directly used in SQL

SECURE UDF and STORED PROCEDURE

```
create or replace role developer_role ;

create or replace user developer1 password = '123'
login_name = 'developer1'
first_name = 'Developer1'
email = ''  -- important to give the email for urgent issues if snowflake wants to connect
default_role = developer_role
must_change_password = false ;

grant role developer_role to user developer1 ;

show grants to user developer1 ;

-- grants on database and schema to developer_role
grant all privileges on database training  to developer_role;
grant all privileges on schema demo to developer_role;


grant all privileges on database training  to developer_role;
grant all privileges on schema demo to developer_role;

grant all privileges on table client to developer_role ;
grant all privileges on table client_stg to developer_role ;
grant all privileges on table client_tgt to developer_role ;

grant usage on function fn_area_of_triangle(float, float) to developer_role ;
grant usage on function fn_active_client(varchar) to developer_role ;
```

developer1

```
-- Login to developer1 and run the below
show functions like 'fn%';
--Describe
describe function fn_area_of_triangle(float, float) ;
describe function fn_active_client(varchar);
-- using DDL
select get_ddl('function', 'fn_area_of_triangle(float, float)');
select get_ddl('function', 'fn_active_client(varchar)');
```

-- making it Secure either while creating OR via ALTER post creation

```
alter function if exists fn_active_client(varchar) set secure  ;
alter function if exists fn_area_of_triangle(float, float) set secure  ;

show functions like 'fn%';
```

Again login to developer1 and verify

-- PROCEDURES

```
grant usage on procedure prc_load_client_tgt() to developer_role ;
```

developer1

```
--show procedure
show procedures like 'prc%';
--describe
describe procedure PRC_LOAD_CLIENT_TGT();
-- get_ddl
select get_ddl('procedure', 'prc_load_client_tgt()');
```

accountadmin

```
-- make it secure

-- can be made only while creating the procedure
drop procedure prc_load_client_tgt() ;

 create or replace secure procedure prc_load_client_tgt()
  returns varchar
  language sql
  as
  $$
    begin
      insert into client_tgt
         select * from client_stg ;

      return 'Record Inserted' ;

   end;
  $$
  ;

-- test the procedure
select * from client_stg ;
select * from client_tgt ; -- empty, if not clean
delete from client_tgt ;
call prc_load_client_tgt() ;
```

```
select * from client_tgt ; -- procedure works

--show procedure
show procedures like 'prc%'; -- check is_secure
--
grant usage on procedure prc_load_client_tgt() to developer_role ;
```

developer1

```
--show procedure
show procedures like 'prc%';
--describe
describe procedure PRC_LOAD_CLIENT_TGT();
-- get_ddl
select get_ddl('procedure', 'prc_load_client_tgt()');
```

VIEWS

1. Views

   It is a DB object which allows the subset of rows OR columns along with dynamic column calculations to be accessed as a table.

   Demo:

   ```
   show tables ;

   select * from client ;
   ```

```
create or replace view v_active_client as select * from client where status = 'ACTIVE' ; -- subset of rows
create or replace view v_client as select id, first_name from client ; --only few columns
create or replace view v_client_extra_cols as
select cl.first_name ||' '|| cl.last_name as full_name, cl.* from client cl ;

--If there is a large number of clients. Different Views can be created to modularize the data
create or replace view v_client_active as select * from client where status = 'ACTIVE' ;
create or replace view v_client_inactive as select * from client where status = 'INACTIVE' ;

--Can be defined as secure as well
create or replace secure view v_cleint_sec as select * from client ;

-- As we have seen in the last session that when you create a secure view, the body is not accessible to user it is shared

show views ;
```

Benefits

Modular code
Table subset access

Limitation:

Read Only
Can not be altered
Table changes not propagated to views

2. View Types

a. Views as explained above also called Non-Materialized Views
b. Materialized Views

3. Materialized Views

a. It is a View which behaves like a table and results are stored as in a table. This helps the faster execution of a frequently needed complicated query.
b. Good use case if the complicated query needs to be called frequently
c. Use select statements
d. Refreshed when you select the data from materialized view.

Demo:

Create materialized view
Alter materialized view
Drop materialized view
Describe materialized view
Show materialized views

INSERT/UPDATE/DELETE are **NOT** allowed on materialized views.

A user should have create materialized view grant to create the materialized view

Grant create materialized view on schema <> to role <>

```
--Demo

-- create or replace materialized view

-- to check the performance of a materialized biew we will disable the caching

alter session set USE_CACHED_RESULT =FALSE ;

create table customer_address as
select * from sample_data.tpcds_sf100tcl.customer_address ; -- takes 28 seconds

-- Lets create a little analytical OR complicated columns query
-- which might take some extra time when doing it at run time

-- We will add some max, min, -- some aggregate functions allowed in Materialized views

select ca_address_id,
       count(distinct ca_zip) as num_zips,
       avg(ca_gmt_offset) as avg_gmt_offset,
       count(*) num_address,
       max(ca_suite_number) max_suite,
       min(ca_location_type) min_loc,
       sum(ca_gmt_offset) sum_g1_offset
from customer_address
group by ca_address_id
;
-- The above query is taking approximately 35 seconds


-- Lets create a materialized view
create or replace materialized view mv_customer_address
as
select ca_address_id,
  --     count(distinct ca_zip) as num_zips,
       approx_count_distinct(distinct ca_zip) as num_zips,
       avg(ca_gmt_offset) as avg_gmt_offset,
       count(*) num_address,
       max(ca_suite_number) max_suite,
       min(ca_location_type) min_loc,
       sum(ca_gmt_offset) sum_g1_offset
from customer_address
group by ca_address_id
```

```
;

-- understand that all aggregate functions are not allowed. Hence I have to comment the count(distinct)

-- Lets select from the view

select * from mv_customer_address ; -- came back in ~ 19-25 seconds versus ~35 seconds above

-- So if you run this frequently this is more efficient.


-- let see materialized views

show materialized views

-- check refreshed on, behind by - 0 means latest

--Let’s check the underlying table changes for data
--Do some DML – delete and updates and inserts

select * from customer_address
where ca_address_id in ('AAAAAAAAAAFGGJCA',
'AAAAAAAAOKEHALBA',
'AAAAAAAAMAAIGJCA',
'AAAAAAAAMHMHALBA',
'AAAAAAAABOGBDPAA',
'AAAAAAAAHGNNPLAA') ;

update customer_address set ca_suite_number = 'Z789299'
where ca_address_id in ('AAAAAAAAAAFGGJCA',
'AAAAAAAAOKEHALBA',
'AAAAAAAAMAAIGJCA',
'AAAAAAAAMHMHALBA',
'AAAAAAAABOGBDPAA',
'AAAAAAAAHGNNPLAA') ;


-- Let's check the query it should show the data
select ca_address_id,
      count(distinct ca_zip) as num_zips,
      avg(ca_gmt_offset) as avg_gmt_offset,
      count(*) num_address,
      max(ca_suite_number) max_suite,
      min(ca_location_type) min_loc,
      sum(ca_gmt_offset) sum_g1_offset
from customer_address where ca_address_id in ('AAAAAAAAAAFGGJCA',
'AAAAAAAAOKEHALBA',
'AAAAAAAAMAAIGJCA',
'AAAAAAAAMHMHALBA',
```

```
'AAAAAAAABOGBDPAA',
'AAAAAAAAHGNNPLAA')
group by ca_address_id
;



-- Check the behind_by -- how far the materialized is behind the updates in the base table
show materialized views;

--check compacted_on -- what time the materialized view was compacted


-- Select from materialized view will refresh the view

select * from mv_customer_address where ca_address_id in ('AAAAAAAAAAFGGJCA',
'AAAAAAAAOKEHALBA',
'AAAAAAAAMAAIGJCA',
'AAAAAAAAMHMHALBA',
'AAAAAAAABOGBDPAA',
'AAAAAAAAHGNNPLAA')  ;

-- check the refresh and compacted by again
show materialized views;

-- First select from MV post DML will be a little longer since it refreshes the view. Subsequent runs will
be quicker.

-- MV is managed by snowflake compute power
-- It doesn't use the Warehouse
-- You can check the cost of maintenance for materialized view
--
   select * from table(information_schema.materialized_view_refresh_history()) ;
-- Not a best case if underlying data is changing too frequently


-- materialized view can be suspended
--suspend MV
alter materialized view mv_customer_address suspend ;

show materialized views ;
--select from MV
select * from mv_customer_address ;
--resume
alter materialized view mv_customer_address resume ;
--
show materialized views ;
-- check the data now
select * from mv_customer_address ;
```

```
-- materialized view can be clustered like a table can be clustered
show materialized views ;
alter materialized view mv_customer_address cluster by (ca_address_id) ;

-- remove clustering
alter materialized view mv_customer_address drop clustering key ;



--- cleanup
drop table customer_address ;
drop materialized view mv_customer_address ;
```

When to Choose MV

1. For frequent complicated query where underlying data is not changing
2. if the data is changing on a regular basis other tasks and streams can be created to handle
3. Think of maintenance cost, refresh costs

Limitation of Materialized Views

1. Enterprise or High ~ MV
2. Joins and Self Joins are not supported
3. Only limited list of aggregate function
4. No UDF's, No Limit Clause, No Having Clause, No Order by clause, No group by, No group by rollups, cube

Data Loading in Snowflake

Bulk Load	Continuous Load
Commonly used method	Designed for micro-batching of data files
Use COPY command	Serverless feature Snowpipe is used.
Load from stages	Loads data automatically once available
Use compute power of Virtual warehouse	100-250MB data files is compresses is optimal

Data can be transformed while loading	

Stages

Snowflake refers to the location of data files in cloud storage as a *stage*

External Stage	Internal Stage ~ snowflake maintained
Database object in schema CREATE STAGE - stores URL, and access settings	a. User @~ b. Table @% c. Named (stores data files within

Named (stores references to file location) Supports a. S3 b. GCP c. Azure	snowflake) @ CREATE STAGE (encryption - FULL or SSE) SNOWFLAKE_FULL : PUT, Already encrypted(Client) SNOWFLAKE_SSE : Later Encrypted (Server)

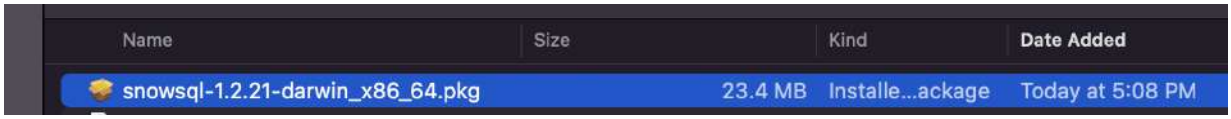
HANDS ON - SNOWSQL

Install SNOWSQL

Download latest version

https://sfc-repo.snowflakecomputing.com/snowsql/bootstrap/1.2/darwin_x86_64/index.html

Double click and the installer will install it.



Modify Paths

https://bigdatadave.com/2020/07/28/snowflake-command-not-found-snowsql/#:~:text=Step%201%3A%20Open%20the%20Terminal,Y%20and%20ENTER%20to%20save.

1. Open Terminal
cd ..
cd ..
ls
cd etc
sudo nano paths
2. Enter Password
3. Add in the path as follows for me:

/Applications/SnowSQL.app/Contents/MacOS/
control + x
Y
[ENTER]

Change the config file

cd ~/.snowsql
Vi config

modify this line:
log_file = ../snowsql_rt.log
to this:
log_file = ~/.snowsql/snowsql_rt.log

https://JF90184.us-east-2.aws.snowflakecomputing.com/console/login.
https://app.snowflake.com/us-east-2.aws

Connect

```
snowsql -a <locator>.us-east-2.aws -u <user> -d <database> -s <schema>


snowsql -a jf90184.us-east-2.aws -u vsingh -d training -s stages

--Table Creation via SNOWSQL
drop table first_table ;
create table first_table (first_column string) ;
--
delete from first_table ;

insert into first_table values ('Test1') ;
insert into first_table values ('Test2') ;
insert into first_table values ('Test3') ;
insert into first_table values ('Test4') ;
insert into first_table values ('Test5') ;
insert into first_table values ('Test6') ;
insert into first_table values ('Test7') ;
commit;

select * from first_table ;

drop table first_table ;
```

Select * from first_table ;

Create Internal Stages - User, Table and Named

-- **User stage**

-- Show stages
show stages

-- Run it via snowsql. User Stage -- can not be altered or drop, not a DB object, do not support set of file format -

-- only at copy time

-- Good use case if a user wants to load the data to multiple table from local files

```
-- Run via snowsql - before that lets see the file
put file:///Users/vipinsingh/Desktop/Snowflake Course/Data/subjects.csv @~/vsingh;
put file:///Users/vipinsingh/Desktop/SnowflakeCourse/Data/UserStageFile/subjects.csv @~/vsingh;


--list
list @~/vsingh;

-- see the data in stage
select t.$1, t.$2, t.$3, t.$4 from @~/vsingh t;

select * from @~/vsingh t; -- doesn't work, works in table stage


-- Create the table for user stage data
create table subjects (name string, sub1 string, sub2 string, sub3 string) ;

-- check the table
select * from subjects ;

-- Copy the data
-- can give file format option here, if file type is not default csv
copy into subjects from @~/vsingh ;

-- select the data
select * from subjects ;

-- Cleanup
drop table subjects ;
remove @~/vsingh pattern='.*.csv.gz' ; -- remove files for re-run – can’t drop user stage

list @~/vsingh ;
```

-- **Table Stage**

```
create table salary (Age number , Sal number) ;

--Via snowsql, table stage should have same name as table, can not be altered or dropped, not a DB object
--Use this if the requirement is to load files in single table
-- Adding the file to table stage, lets see the simple file

-- Run it via snowsql
put file:///Users/vipinsingh/Desktop/SnowflakeCourse/Data/TableStageFile/salary.csv @%salary;
```

```
--check the records in table_stage
select * from salary ;

--list the files in table stage
list @%salary;

-- Select specific columns from table stage
select t.$1, t.$2 from @%salary t;

-- Can directly query the tabe stage as well
select * from @%salary t; -- direct select * works in table stages

--use copy command to load the data in to table_stage
-- can specify file format option here if file type is not csv
copy into salary from @%salary ;

-- Validate the rows in tabe_stage
select * from salary ;

--cleanup
drop table salary ; -- dropping table will remove the data from stage
-- OR
remove @%salary pattern='.*.csv.gz' ; -- to reload, we can remove the files.

select * from @%salary t;
```

Internal Stage - Named

```
-- create internal stage the name is ‘shared_stage’’
create stage shared_stage ;

-- show all stages
show stages ;

-- Describe internal stage
desc stage shared_stage  ;

-- Alter internal stage
-- good use case to load files via multiple users to multiple tables
```

```
alter stage shared_stage rename to shared_stage2;

-- Describe
desc shared_stage2 ;

alter stage shared_stage2 rename to shared_stage;

-- Run it via snowsql, use case for multiple users loading multiple tables.
-- check the file
put file:///Users/vipinsingh/Desktop/SnowflakeCourse/Data/InternalNamedStageFile/weather.csv @shared_stage;


--select columns from internal named stage
select t.$1, t.$2  from @shared_stage t;

select * from @shared_stage t; – Works for table stage, You need to specify columns


-- create table
drop table weather ;

-- create a table to load internal named stage data
create table weather (city string, temperature number);

-- Copy the data from internal named stage to table
copy into weather from @shared_stage ;

-- Check the data in the table
select * from weather ;

-- Cleanup
drop stage shared_stage ;
drop table weather ;
```

INTERNAL STAGES

User Stage	Table Stage	Named Stage
Use Case: If a user wants to load the data from a local file to multiple tables	Use Case: Specific to table, if the file has to be loaded to a single table.	Use Case: If multiple users wants to load files to multiple tables

External Stages - Named

```
-- Creating external stage for publicly available file

CREATE OR REPLACE STAGE contacts_stage url='s3://snowflake-docs/tutorials/dataloading/' ;

DESC STAGE contacts_stage;

SHOW stages ;

LIST @contacts_stage;

ALTER STAGE contacts_stage rename to contacts_stage1;

DESC STAGE contacts_stage1;

ALTER STAGE contacts_stage1 rename to contacts_stage;


create or replace table CONTACTS (
  id integer,
  last_name string,
  first_name string,
```

```
  company string,
  email string,
  workphone string,
  cellphone string,
  streetaddress string,
  city string,
  postalcode string);

  COPY INTO CONTACTS
    FROM @contacts_stage
    file_format = (          type = 'CSV'
                             field_delimiter = '|'
                             skip_header = 1)
    pattern='.*.contacts5.csv';


select * from CONTACTS ;

drop table CONTACTS ;

drop stage contacts_stage ;
```

COPY Command

```
--Create external stage pointing to files at AWS location
CREATE OR REPLACE STAGE ext_aws_stage url='s3://snowflake-docs/tutorials/dataloading/' ;

--List the Stage. It shows the files.
list @ext_aws_stage ;

--Create a table to load the files. The tables should have exact number of columns as files,
create or replace table CONTACTS (
  id integer,
  last_name string,
  first_name string,
  company string,
  email string,
  workphone string,
  cellphone string,
```

```
  streetaddress string,
  city string,
  postalcode string);

--delete for rerun, doesn't work
delete from CONTACTS ;
commit;

--truncate fir rerun
truncate table CONTACTS ;

--select from table
select * from contacts ;



-- We used a copy here. Let's discuss in detail. Exact file.
-- If you are working in the same database/schema then no need to qualify the external stage, otherwise – qualify
-- Try Running with and without the files clause and observe the error.

COPY INTO CONTACTS FROM @ext_aws_stage
file_format = (    type = 'CSV'
                                field_delimiter = '|'
                                skip_header = 1
            );

--List the Stage. It shows the files.
list @ext_aws_stage ;

-- Give the Files to load specific file
COPY INTO CONTACTS FROM @ext_aws_stage
file_format = (    type = 'CSV'
                                field_delimiter = '|'
                                skip_header = 1
            )
files=('contacts1.csv');

--  Load the file contacts1 successfully. Try rerunning the command.
select * from CONTACTS ;

-- We used a copy here. Let's discuss in detail. Exact two files.
-- How can you choose exact two files to load

list @ext_aws_stage ;


--If you want to load more than one specific file. You can give it,
COPY INTO CONTACTS FROM @ext_aws_stage
```

```
file_format = (    type = 'CSV'
                                field_delimiter = '|'
                                skip_header = 1
            )
   files=('contacts2.csv','contacts4.csv');

select * from CONTACTS ;


-- Copy with pattern
list @ext_aws_stage ;
truncate table contacts ;

COPY INTO CONTACTS FROM @ext_aws_stage
file_format = (    type = 'CSV'
                                field_delimiter = '|'
                                skip_header = 1
            )
   pattern='.*contacts[1-2].csv' ;

select * from CONTACTS ;


--cleanup
drop stage ext_aws_stage ;
drop table CONTACTS ;
```

```
--Create external stage pointing to files at AWS location
CREATE OR REPLACE STAGE ext_aws_stage url='s3://snowflake-docs/tutorials/dataloading/' ;

--List the Stage. It shows the files.
list @ext_aws_stage ;


--Create a table to load the files. The tables should have exact number of columns as files,
create or replace table CONTACTS (
  id integer,
  last_name string,
  first_name string,
  company string,
  email string,
  workphone string,
  cellphone string,
  streetaddress string,
  city string,
  postalcode string);

--Copy and observe the data
COPY INTO CONTACTS FROM @ext_aws_stage
file_format = (    type = 'CSV'
                                    field_delimiter = '|'
                                    skip_header = 1
             )
files=('contacts5.csv');

-- Check the data

select * from contacts ;


-- create a table having different columns than the stage. Hence File columns and table columns can be
different
create table contacts_transformation1 (id number, last_name string ) ;

-- Transformation
    --specific columns can be selected or some columns can be omitted.
    --column ordering can be changed
    --casting can be done at column level


-- Example 1 - specific columns to table from the file

COPY INTO contacts_transformation1
    FROM (select s.$1, s.$2 from @ext_aws_stage s)
```

```
    file_format = (         type = 'CSV'
                            field_delimiter = '|'
                            skip_header = 1)
    pattern='.*.contacts5.csv';


select * from contacts_transformation1 ;


-- Example 2 - Column reordering
create table contacts_transformation2 (first_name string, last_name string ) ;

COPY INTO contacts_transformation2
    FROM (select s.$3, s.$2 from @ext_aws_stage s)
    file_format = (         type = 'CSV'
                            field_delimiter = '|'
                            skip_header = 1)
    pattern='.*.contacts5.csv';


select * from contacts_transformation2 ;

-- Example 3 - Transformation using a function
create table contacts_transformation3(first_name string, last_name string, full_name string) ;

COPY INTO contacts_transformation3
    FROM (select s.$3, s.$2, concat(s.$3,' ',s.$2) from @ext_aws_stage s)
    file_format = (         type = 'CSV'
                            field_delimiter = '|'
                            skip_header = 1)
    pattern='.*.contacts5.csv';


select * from contacts_transformation3 ;


-- Example 4 - Not all function are supported - example lower
create table contacts_transformation4(first_name string, last_name string, lower_full_name string) ;

COPY INTO contacts_transformation4
    FROM (select s.$3, s.$2, lower(concat(s.$3,' ',s.$2)) from @ext_aws_stage s)
    file_format = (         type = 'CSV'
                            field_delimiter = '|'
                            skip_header = 1)
    pattern='.*.contacts5.csv';

    -- error
    -- There is a specific list of transformation function are supported in COPY
```

```
select * from contacts_transformation4 ;


-- Example 5 - Change types
create table contacts_transformation5(first_name string, last_name string, bin_first_name binary) ;

COPY INTO contacts_transformation5
   FROM (select s.$3, s.$2, to_binary(s.$3,'utf-8') from @ext_aws_stage s)
   file_format = (          type = 'CSV'
                            field_delimiter = '|'
                            skip_header = 1)
   pattern='.*.contacts5.csv';


select * from contacts_transformation5 ;


-- Example 6 - Sequence column in table
create sequence id_seq ;

create table contacts_transformation6(id number default id_seq.nextval, first_name string, last_name
string) ;

COPY INTO contacts_transformation6
   FROM (select id_seq.nextval, s.$3, s.$2 from @ext_aws_stage s)
   file_format = (          type = 'CSV'
                            field_delimiter = '|'
                            skip_header = 1)
   pattern='.*.contacts5.csv';


select * from contacts_transformation6 ;


-- Example 7 - Autoincrement

create table contacts_transformation7(idai number autoincrement start 99999 increment 10000,
first_name string, last_name string) ;

COPY INTO contacts_transformation7 (first_name, last_name)
   FROM (select s.$3, s.$2 from @ext_aws_stage s)
   file_format = (          type = 'CSV'
                            field_delimiter = '|'
                            skip_header = 1)
   pattern='.*.contacts5.csv';


select * from contacts_transformation7 ;
```

```
-- Cleanup
drop stage ext_aws_stage ;
drop table contacts ;
drop table contacts_transformation1 ;
drop table contacts_transformation2 ;
drop table contacts_transformation3 ;
drop table contacts_transformation4 ;
drop table contacts_transformation5 ;
drop table contacts_transformation6 ;
drop table contacts_transformation7 ;
drop sequence id_seq ;
```

File Format Object

```
-- File Format Object

-- Creating contacts_stage for publicly available file
CREATE OR REPLACE STAGE ext_aws_stage url='s3://snowflake-docs/tutorials/dataloading/' ;

-- Description of contacts_stage
DESC STAGE ext_aws_stage;

--list
LIST @ext_aws_stage;

--
drop table contacts ;
--
--Create a table to load the files. The tables should have exact number of columns as files,
create or replace table CONTACTS (
  id integer,
  last_name string,
  first_name string,
  company string,
  email string,
  workphone string,
  cellphone string,
  streetaddress string,
  city string,
  postalcode string);
```

```
-- File format in the copy command
 COPY INTO contacts
   FROM @ext_aws_stage
   file_format = (          type = 'CSV'
                            field_delimiter = '|'
                            skip_header = 1)
   pattern='.*.contacts5.csv';

--check the data
select * from contacts ;

-- We can create file format as schema objects
create or replace file format contacts_file_format;

-- check the properties of file format
desc file format contacts_file_format;

-- Truncate
truncate table contacts ;

-- Using file format object below
   copy into contacts
   from @ext_aws_stage
   file_format = (format_name = contacts_file_format)
   pattern='.*.contacts5.csv';

-- Check the table
select * from contacts ;

-- We can alter file format object
alter file format contacts_file_format set skip_header  = 1;
alter file format contacts_file_format set field_delimiter  = '|';

-- Check again
desc file format contacts_file_format ;

-- Run above COPY again

-- check data again
select * from contacts


-- ALTER FILE FORMAT - Few things to Note

--1 Can alter name
alter file format if exists contacts_file_format rename to new_contacts_file_format;
show file formats ;
```

```
--2 Can alter comments. Use show file format command to see comments.
alter file format new_contacts_file_format set comment = 'test the comment' ;
show file formats ;

-- Check
desc file format new_contacts_file_format ;
--3 Can alter formattypeoptions for a Type.We saw above
alter file format new_contacts_file_format set skip_header  = 1;
--Note: Can not alter TYPE, You need to recreate the file format.
alter file format new_contacts_file_format set type  = 'JSON';

--rename back
alter file format if exists new_contacts_file_format rename to contacts_file_format;

-- recreate
create or replace file format contacts_file_format;

-- Overwriting properties of file format object
truncate table contacts ;

 copy into contacts
    from @ext_aws_stage
    file_format = (format_name = contacts_file_format field_delimiter = '|' skip_header=1 )
    pattern='.*.contacts5.csv';

-- select
select * from contacts ;

-- Properties of stage is a superset, all properties of file format are in stage
desc stage ext_aws_stage;
desc file format contacts_file_format ;

-- cleanup
drop stage if exists ext_aws_stage ;
drop file format contacts_file_format ;
drop table contacts ;
```

COPY OPTIONS

```
/* Standard data load */
COPY INTO [<namespace>.]<table_name>
     FROM { internalStage | externalStage | externalLocation }
[ FILES = ( '<file_name>' [ , '<file_name>' ] [ , ... ] ) ]
[ PATTERN = '<regex_pattern>' ]
[ FILE_FORMAT = ( { FORMAT_NAME = '[<namespace>.]<file_format_name>' |
                    TYPE = { CSV | JSON | AVRO | ORC | PARQUET | XML } [ formatTypeOptions ] } ) ]
[ copyOptions ]
[ VALIDATION_MODE = RETURN_<n>_ROWS | RETURN_ERRORS | RETURN_ALL_ERRORS ]
```

```
copyOptions ::=
     ON_ERROR = { CONTINUE | SKIP_FILE | SKIP_FILE_<num> | 'SKIP_FILE_<num>%' | ABORT_STATEMENT }
     SIZE_LIMIT = <num>
     PURGE = TRUE | FALSE
     RETURN_FAILED_ONLY = TRUE | FALSE
     MATCH_BY_COLUMN_NAME = CASE_SENSITIVE | CASE_INSENSITIVE | NONE
     ENFORCE_LENGTH = TRUE | FALSE
     TRUNCATECOLUMNS = TRUE | FALSE
     FORCE = TRUE | FALSE
     LOAD_UNCERTAIN_FILES = TRUE | FALSE
```

```
-- Creating contacts_stage for publicly available file
create or replace stage ext_aws_stage url='s3://snowflake-docs/tutorials/dataloading/' ;

-- Description
desc stage ext_aws_stage;

--list
list @ext_aws_stage;

--Create Table
create or replace table CONTACTS (
  id integer,
  last_name string,
  first_name string,
  company string,
  email string,
  workphone string,
  cellphone string,
  streetaddress string,
  city string,
  postalcode string);
```

```
--  copy command
 COPY INTO contacts
    FROM @ext_aws_stage
    file_format = (          type = 'CSV'
                             field_delimiter = '|'
                             skip_header = 1)
    files = ('contacts1.csv','contacts2.csv','contacts3.csv');


 -- check data
select * from contacts ;

-- Truncate Table
truncate table contacts ;

--  Handle Error using ON_ERROR
 COPY INTO contacts
    FROM @ext_aws_stage
    file_format = (          type = 'CSV'
                             field_delimiter = '|'
                             skip_header = 1)
    files = ('contacts1.csv','contacts2.csv','contacts3.csv')
    on_error = 'CONTINUE';

  -- Validate data and count
select * from contacts order by 1;

--truncate the table
truncate table contacts ;

-- By default in our first copy ABORT_STATEMENT was applied. If we don't give on_error option it is abort statement, Lets see the
-- the result by specifying explicitly.
 COPY INTO contacts
    FROM @ext_aws_stage
    file_format = (          type = 'CSV'
                             field_delimiter = '|'
                             skip_header = 1)
    files = ('contacts1.csv','contacts2.csv','contacts3.csv')
    on_error = 'ABORT_STATEMENT';


  -- Validate data and count
select * from contacts order by 1;

--Truncate before re-running copy to clean up metadata information
truncate table contacts ;
```

```
-- Handling Error Using  = SKIP_FILE
COPY INTO contacts
   FROM @ext_aws_stage
   file_format = (          type = 'CSV'
                            field_delimiter = '|'
                            skip_header = 1)
   files = ('contacts1.csv','contacts2.csv','contacts3.csv')
   on_error = 'SKIP_FILE';

-- Validate data
select * from contacts order by 1;

--Truncate before re-running copy to clean up metadata information
truncate table contacts ;

-- Handling error using SKIP_FILE_<error_number>
COPY INTO contacts
   FROM @ext_aws_stage
   file_format = (          type = 'CSV'
                            field_delimiter = '|'
                            skip_header = 1)
   files = ('contacts1.csv','contacts2.csv','contacts3.csv')
   on_error = 'SKIP_FILE_2';


  -- Validate data and count
select * from contacts order by 1;

--Truncate before re-running copy to clean up metadata information
truncate table contacts ;

-- Error handling using the ON_ERROR option = SKIP_FILE_<percent_numbe>%
COPY INTO contacts
   FROM @ext_aws_stage
   file_format = (          type = 'CSV'
                            field_delimiter = '|'
                            skip_header = 1)
   files = ('contacts1.csv','contacts2.csv','contacts3.csv')
   on_error = 'SKIP_FILE_60%';


  -- Validate data and count
select * from contacts order by 1;

--Cleanup
drop stage if exists ext_aws_satge ;
drop table contacts ;
```

SIZE_LIMIT = <num>

1. > 0, max size (**in bytes**) of data to be loaded for a COPY command.
2. Default null , no size limit.
3. Used to control the COPY statement - How much data it should load
4. Atleast ONE file is loaded regardless of size.

Hands On

```
-- Creating contacts_stage for publicly available file
create or replace stage ext_aws_stage url='s3://snowflake-docs/tutorials/dataloading/' ;

-- Description
desc stage ext_aws_stage;

--list
list @ext_aws_stage;

--Create Table
create or replace table CONTACTS (
  id integer,
  last_name string,
```

```
  first_name string,
  company string,
  email string,
  workphone string,
  cellphone string,
  streetaddress string,
  city string,
  postalcode string);

-- Truncate Table
truncate table contacts ;


--  COPY Command
 COPY INTO contacts
    FROM @ext_aws_stage
    file_format = (          type = 'CSV'
                             field_delimiter = '|'
                             skip_header = 1)
    files = ('contacts1.csv','contacts2.csv','contacts4.csv','contacts5.csv')
    size_limit = 500;

--  COPY Command
 COPY INTO contacts
    FROM @ext_aws_stage
    file_format = (          type = 'CSV'
                             field_delimiter = '|'
                             skip_header = 1)
    files = ('contacts1.csv','contacts2.csv','contacts4.csv','contacts5.csv')
    size_limit = 1300;

-- COPY
COPY INTO contacts
    FROM @ext_aws_stage
    file_format = (          type = 'CSV'
                             field_delimiter = '|'
                             skip_header = 1)
    files = ('contacts1.csv','contacts2.csv','contacts4.csv','contacts5.csv')
    size_limit = 1300 ;

 -- check data
select * from contacts ;


  -- Validate data and count
select * from contacts order by 1;

--truncate the table
truncate table contacts ;
```

```
-- Cleanup
drop table contacts ;
drop stage ext_aws_stage ;
```

RETURN_FAILED_ONLY = TRUE | FALSE

- Used with ON_ERROR
- To specify to return the files that have failed
- By Default it is FALSE

Hands On

```
-- Creating contacts_stage for publicly available file
create or replace stage ext_aws_stage url='s3://snowflake-docs/tutorials/dataloading/' ;

-- Description
desc stage ext_aws_stage;

--list
list @ext_aws_stage;

--Create Table
create or replace table CONTACTS (
  id integer,
  last_name string,
  first_name string,
  company string,
  email string,
  workphone string,
  cellphone string,
  streetaddress string,
  city string,
  postalcode string);
```

```
--  copy command
 COPY INTO contacts
    FROM @ext_aws_stage
    file_format = (          type = 'CSV'
                             field_delimiter = '|'
                             skip_header = 1)
    files = ('contacts1.csv','contacts2.csv','contacts3.csv','contacts4.csv','contacts5.csv')
    on_error = 'CONTINUE';

 -- check data
select * from contacts ;

-- Truncate Table
truncate table contacts ;

 --  copy command with RETURN_FAILED_ONLY
 COPY INTO contacts
    FROM @ext_aws_stage
    file_format = (          type = 'CSV'
                             field_delimiter = '|'
                             skip_header = 1)
    files = ('contacts1.csv','contacts2.csv','contacts3.csv','contacts4.csv','contacts5.csv')
    on_error = 'CONTINUE'
    RETURN_FAILED_ONLY = TRUE;

-- Cleanup
drop stage ext_aws_stage ;
drop table contacts ;
```

ENFORCE_LENGTH = TRUE | FALSE

- Default TRUE
- FALSE will Load and truncate the values to the column width

TRUNCATECOLUMNS = TRUE | FALSE

- Default FALSE
- TRUE will Load and truncate the values to the column width

Why are two Copyoption doing the same thing?

The answer is to be compatible with outer systems syntax.

Hands On

```
-- Creating contacts_stage for publicly available file
create or replace stage ext_aws_stage url='s3://snowflake-docs/tutorials/dataloading/' ;

-- Description
desc stage ext_aws_stage;

--list
list @ext_aws_stage;

--Create Table
create or replace table CONTACTS (
  id integer,
  last_name varchar(3),
  first_name string,
  company string,
  email string,
  workphone string,
  cellphone string,
  streetaddress string,
  city string,
  postalcode string);

--truncate table
truncate table contacts ;

--  copy with enforcelength
 COPY INTO contacts
    FROM @ext_aws_stage
    file_format = (          type = 'CSV'
                             field_delimiter = '|'
                             skip_header = 1)
    files = ('contacts1.csv','contacts2.csv','contacts4.csv','contacts5.csv')
    on_error = 'CONTINUE';
  --  ENFORCE_LENGTH = TRUE;


-- Truncate Table
truncate table contacts ;

-- Copy with truncatecolumns
COPY INTO contacts
    FROM @ext_aws_stage
    file_format = (          type = 'CSV'
                             field_delimiter = '|'
```

```
                                skip_header = 1)
    files = ('contacts1.csv','contacts2.csv','contacts4.csv','contacts5.csv')
    on_error = 'CONTINUE';
    --TRUNCATECOLUMNS = FALSE;

 -- check data
select * from contacts ;

--  copy with enforcelength
 COPY INTO contacts
    FROM @ext_aws_stage
    file_format = (            type = 'CSV'
                                field_delimiter = '|'
                                skip_header = 1)
    files = ('contacts1.csv','contacts2.csv','contacts4.csv','contacts5.csv')
    on_error = 'CONTINUE'
    ENFORCE_LENGTH = FALSE;


-- Truncate Table
truncate table contacts ;

-- Copy with truncatecolumns
COPY INTO contacts
    FROM @ext_aws_stage
    file_format = (            type = 'CSV'
                                field_delimiter = '|'
                                skip_header = 1)
    files = ('contacts1.csv','contacts2.csv','contacts4.csv','contacts5.csv')
    on_error = 'CONTINUE'
    TRUNCATECOLUMNS = TRUE;

-- Cleanup
drop stage ext_aws_stage ;
drop table contacts ;
```

FORCE = TRUE | FALSE

- Default False
- True, loads the file again and again without skipping

LOAD_UNCERTAIN_FILES=TRUE|FALSE

- Default False
- If the file is loaded 64 days or more earlier, Status of file is uncertain in these cases snowflake copy command skips the file load. **Note: Snowflake maintains metadata for 64 days**
- True - will load those files again.

Hands On.

```
-- Creating contacts_stage for publicly available file
create or replace stage ext_aws_stage url='s3://snowflake-docs/tutorials/dataloading/' ;

--list
list @ext_aws_stage;

--Create Table
create or replace table CONTACTS (
  id integer,
  last_name string,
  first_name string,
  company string,
  email string,
  workphone string,
  cellphone string,
  streetaddress string,
  city string,
  postalcode string);


--  copy
 COPY INTO contacts
    FROM @ext_aws_stage
    file_format = (          type = 'CSV'
                             field_delimiter = '|'
                             skip_header = 1)
    files = ('contacts1.csv','contacts2.csv','contacts4.csv','contacts5.csv')
    FORCE = TRUE;


select * from contacts ;

--truncate table
```

```
truncate table contacts ;


--
-- Copy -- 64 days Load metadata for the file expires
 COPY INTO contacts
   FROM @ext_aws_stage
   file_format = (          type = 'CSV'
                            field_delimiter = '|'
                            skip_header = 1)
   files = ('contacts1.csv','contacts2.csv','contacts4.csv','contacts5.csv')
   on_error = CONTINUE
   LOAD_UNCERTAIN_FILES = TRUE;



-- Cleanup
drop stage ext_aws_stage ;
drop table contacts ;





-- Creating contacts_stage for publicly available file
create or replace stage ext_aws_stage url='s3://snowflake-docs/tutorials/dataloading/' ;

--list
list @ext_aws_stage;

--Create Table
create or replace table CONTACTS (
  id integer,
  last_name string,
  first_name string,
  company string,
  email string,
  workphone string,
  cellphone string,
  streetaddress string,
  city string,
  postalcode string);


--  copy
 COPY INTO contacts
   FROM @ext_aws_stage
   file_format = (          type = 'CSV'
```

```
                                field_delimiter = '|'
                                skip_header = 1)
    files = ('contacts1.csv','contacts2.csv','contacts4.csv','contacts5.csv')
    FORCE = TRUE;


select * from contacts ;

--truncate table
truncate table contacts ;


--
-- Copy -- 64 days Load metadata for the file expires
 COPY INTO contacts
    FROM @ext_aws_stage
    file_format = (             type = 'CSV'
                                field_delimiter = '|'
                                skip_header = 1)
    files = ('contacts1.csv','contacts2.csv','contacts4.csv','contacts5.csv')
    on_error = CONTINUE
    LOAD_UNCERTAIN_FILES = TRUE;



-- Cleanup
drop stage ext_aws_stage ;
drop table contacts ;
```

LOAD HISTORY

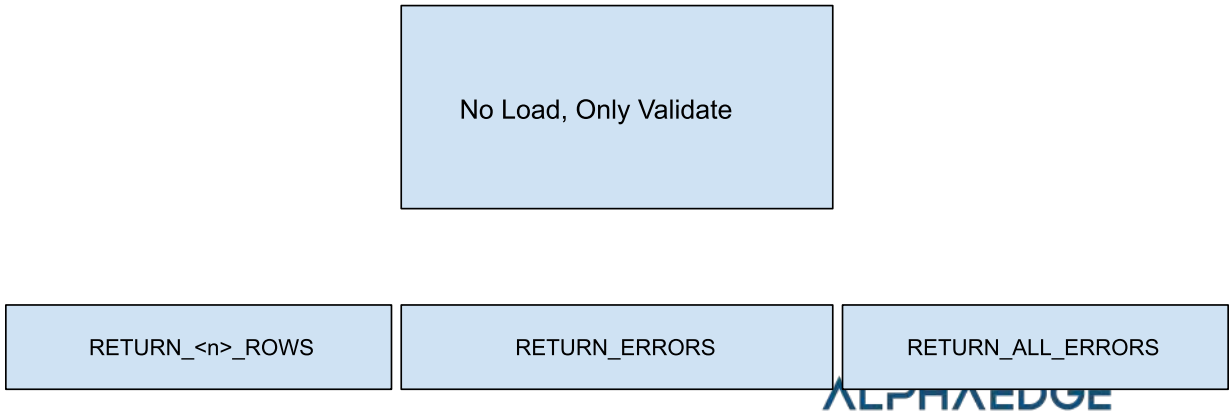
```
-- last 14 days
select * from information_schema.LOAD_HISTORY ;
--schema_name, file_name, table_name,
last_load_time,status,row_count,row_parsed,first_error_message,first_error_line_number,
--first_error_character_position, first_error_col_name,error_count, error_limit

-- Global Level Load History -- 365 days OR 1 year
select * from snowflake.account_usage.LOAD_HISTORY
where schema_name = 'STAGES'
and table_name = 'CONTACTS'
order by 1 desc;

--table_id, table_name,schema_id,schema_name, catalog_id, catalog_name,
file_name,last_load_time,status,row_count,row_parsed,first_error_message,first_error_line_number,
--first_error_character_position, first_error_col_name,error_count, error_limit
```

```
-- Use the global load History to get more information
select * from snowflake.account_usage.load_history where date(last_load_time) = current_date() ;
select * from snowflake.account_usage.load_history where date(last_load_time) < current_date() ;


-- See the errors encountered today
select * from snowflake.account_usage.load_history where date(last_load_time) = current_date() and
error_count > 0;
```

VALIDATION MODE

- The difference between the above two screenshot is Transformation and Validation mode.
- Validation Mode is not allowed when data transformations are done.
- Transformation can not be done in an external location. This is the location on cloud service provider
- externalStage Vs externallocation - One refers to the location and another is external location. Both can be used.

VALIDATION_MODE



Handson - Validation mode.

```
-- Creating external stage for publicly available file
CREATE OR REPLACE STAGE ext_stage url='s3://snowflake-docs/tutorials/dataloading/' ;

-- List the files in external stage
LIST @ext_stage;

-- Create table to demonstrate the validation mode
create table VALIDATION_MODE_CONTACTS as select * from alphaedge_first_db.public.contacts
where 1 = 2;

-- Truncate table to load again if required
truncate table VALIDATION_MODE_CONTACTS ;

-- Use the validation mode, it will not load the data, just return the errors
COPY INTO VALIDATION_MODE_CONTACTS
   FROM @ext_stage
   file_format = (          type = 'CSV'
                            field_delimiter = '|'
                            skip_header = 1)
   files = ('contacts1.csv','contacts2.csv')
   validation_mode = RETURN_ERRORS ;

-- Validate that the data is not loaded.
select * from VALIDATION_MODE_CONTACTS ;

-- contacts3 has errors
COPY INTO VALIDATION_MODE_CONTACTS
   FROM @ext_stage
   file_format = (          type = 'CSV'
                            field_delimiter = '|'
                            skip_header = 1)
   files = ('contacts1.csv','contacts2.csv','contacts3.csv')
   validation_mode = RETURN_ERRORS ;

-- Validate that the data is not loaded.
select * from VALIDATION_MODE_CONTACTS ;
-- Use the validation mode, it will not load the data, just return the rows specified if there are no errors
```

```
COPY INTO VALIDATION_MODE_CONTACTS
   FROM @ext_stage
   file_format = (          type = 'CSV'
                            field_delimiter = '|'
                            skip_header = 1)
   files = ('contacts1.csv','contacts2.csv','contacts3.csv')
   validation_mode = RETURN_7_ROWS ;  -- first 8 rows has at least one error

-- Validate that the data is not loaded.
select * from VALIDATION_MODE_CONTACTS ;

-- RETURN_ERRORS Vs RETURN_ALL_ERRORS

-- truncate table to re-run
truncate table VALIDATION_MODE_CONTACTS ;

-- Load the partial file with errors
COPY INTO VALIDATION_MODE_CONTACTS
   FROM @ext_stage
   file_format = (          type = 'CSV'
                            field_delimiter = '|'
                            skip_header = 1)
   files = ('contacts3.csv')
   on_error = continue ;

-- Validate that the data is not loaded.
select * from VALIDATION_MODE_CONTACTS ;

-- Now run the copy command with validation mode - RETURN_ERRORS Vs RETURN_ALL_ERRORS
COPY INTO VALIDATION_MODE_CONTACTS
   FROM @ext_stage
   file_format = (          type = 'CSV'
                            field_delimiter = '|'
                            skip_header = 1)
   files = ('contacts1.csv','contacts2.csv','contacts3.csv')
   validation_mode = RETURN_ERRORS ;  -- Will not show any errror, only considers files contact1.csv
and contact2.csv


   COPY INTO VALIDATION_MODE_CONTACTS
   FROM @ext_stage
   file_format = (          type = 'CSV'
                            field_delimiter = '|'
                            skip_header = 1)
   files = ('contacts1.csv','contacts2.csv','contacts3.csv')
      validation_mode = RETURN_ALL_ERRORS ;  -- -- Will show error of previously loaded file
contact3.csv
```

```
   COPY INTO VALIDATION_MODE_CONTACTS
   FROM @ext_stage
   file_format = (          type = 'CSV'
                            field_delimiter = '|'
                            skip_header = 1)
   files = ('contacts1.csv','contacts2.csv')
    validation_mode = RETURN_ALL_ERRORS ;  -- -- Will not show all errors if file contact3.csv is not
listed

-- Validate that the dpartial load
select * from VALIDATION_MODE_CONTACTS ;

-- cleanup -- Needed for COPY transformation
drop table VALIDATION_MODE_CONTACTS ;

list @ext_stage ;
drop stage ext_stage ;
```

LOGGING REJECTED RECORDS

- COPY command Shows errors
  - VALIDATION_MODE=RETURN_ERRORS
  - ON_ERROR = CONTINUE

- How can we really log, understand and analyze the rejected records?

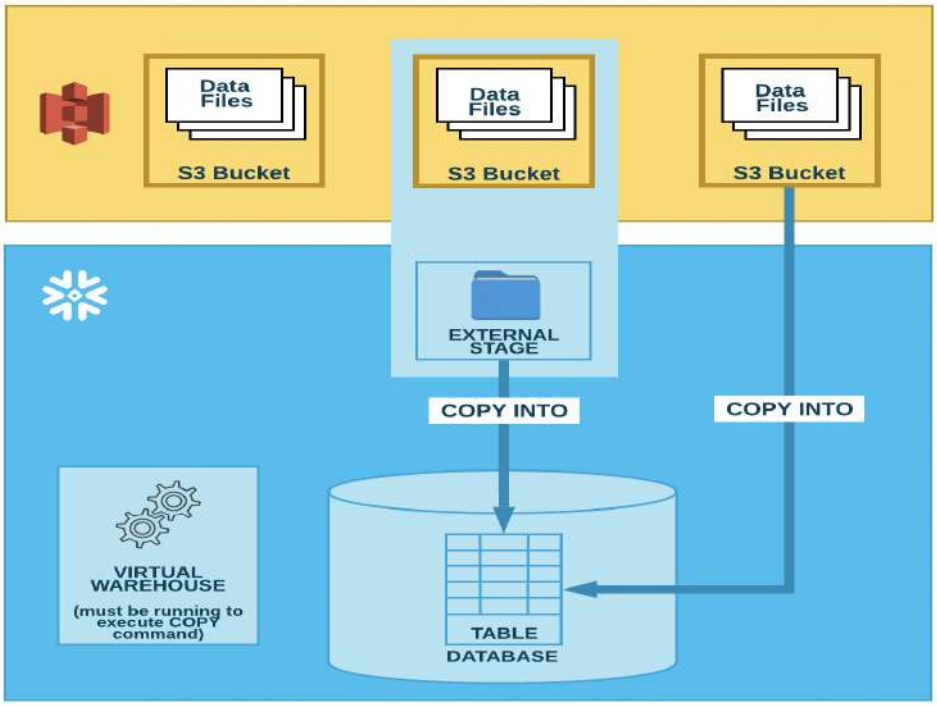
```
-- Creating contacts_stage for publicly available file
create or replace stage ext_aws_stage url='s3://snowflake-docs/tutorials/dataloading/' ;

--list
list @ext_aws_stage;

--Create Table
create or replace table CONTACTS (
  id integer,
  last_name string,
  first_name string,
  company string,
  email string,
  workphone string,
  cellphone string,
  streetaddress string,
```

```
 city string,
 postalcode string);


  --Create rejected table
create or replace table CONTACTS_REJECTED (error string, file string, rejected_record string);

 -- truncate
 truncate table contacts ;

 --copy with errors usimg Validation mode
COPY INTO contacts
   FROM @ext_aws_stage
   file_format = (          type = 'CSV'
                            field_delimiter = '|'
                            skip_header = 1)
   files = ('contacts3.csv')
   VALIDATION_MODE = RETURN_ERRORS;

-- Returns the result of the previous command with in 24 hrs

-- Check the rejected records
insert into CONTACTS_REJECTED select error, file, rejected_record from
table(result_scan(last_query_id()));

commit ;
--Rejected Record can be seen by queryID
select error, file, rejected_record from table(result_scan('01a616cb-0000-b1da-0002-99e60002c216'));

select * from contacts_rejected ;
--Splitted Rejected record
select error, file,
      split_part(rejected_record,'|',1) as ID,
      split_part(rejected_record,'|',2) as LAST_NAME,
      split_part(rejected_record,'|',3) as FIRST_NAME,
      split_part(rejected_record,'|',4) as COMPANY,
      split_part(rejected_record,'|',5) as EMAIL,
      split_part(rejected_record,'|',6) as WORK_PHONE,
      split_part(rejected_record,'|',7) as CELL_PHONE,
      split_part(rejected_record,'|',8) as STREET_ADDRESS,
      split_part(rejected_record,'|',9) as CITY,
      split_part(rejected_record,'|',10) as POSTAL_CODE,
      split_part(rejected_record,'|',11) as OTHER1,
      split_part(rejected_record,'|',12) as OTHER2,
      split_part(rejected_record,'|',13) as OTHER3
from CONTACTS_REJECTED ;
```

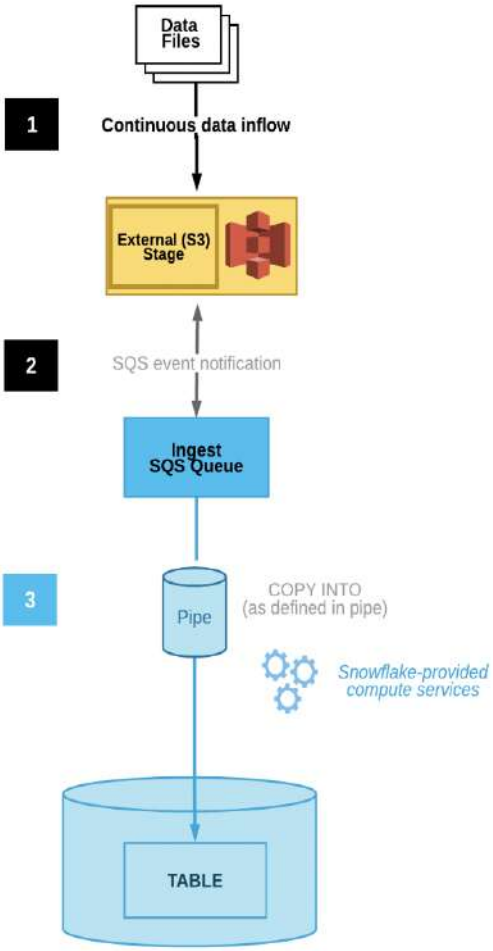
```
-- truncate
truncate table contacts ;

truncate table contacts_rejected ;

--copy with errors using on_error continue
 COPY INTO contacts
   FROM @ext_aws_stage
   file_format = (          type = 'CSV'
                            field_delimiter = '|'
                            skip_header = 1)
   files = ('contacts3.csv')
   ON_ERROR = CONTINUE;

 select * from table(validate(contacts,job_id => '_last')) ;
 select * from table(validate(contacts,job_id => '01a616d1-0000-b1da-0002-99e60002c256')) ;

-- Check the rejected records
insert into CONTACTS_REJECTED select error, file, rejected_record from
table(validate(contacts,job_id => '01a616d1-0000-b1da-0002-99e60002c256')) ;

commit ;

--Splitted Rejected record
select error, file,
      split_part(rejected_record,'|',1) as ID,
      split_part(rejected_record,'|',2) as LAST_NAME,
      split_part(rejected_record,'|',3) as FIRST_NAME,
      split_part(rejected_record,'|',4) as COMPANY,
      split_part(rejected_record,'|',5) as EMAIL,
      split_part(rejected_record,'|',6) as WORK_PHONE,
      split_part(rejected_record,'|',7) as CELL_PHONE,
      split_part(rejected_record,'|',8) as STREET_ADDRESS,
      split_part(rejected_record,'|',9) as CITY,
      split_part(rejected_record,'|',10) as POSTAL_CODE,
      split_part(rejected_record,'|',11) as OTHER1,
      split_part(rejected_record,'|',12) as OTHER2,
      split_part(rejected_record,'|',13) as OTHER3
from CONTACTS_REJECTED ;

-- cleanup
drop table contacts ;
drop table contacts_rejected ;
drop stage ext_aws_stage ;
```

DATA LOADING BEST PRACTICES

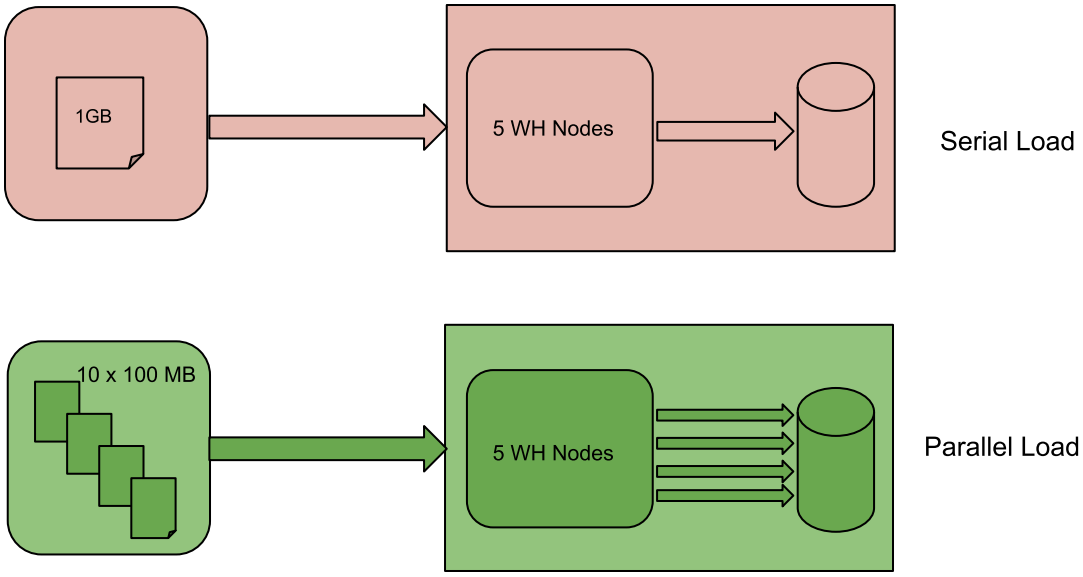
- Prefer BULK Loading
  - Snowflake is designed for bulk load
  - Each Insert/update/delete creates new micro-partition
  - Small micropartition are inefficient for data processing



- Use Snow Pipe for Micro - Batching
    - If the files are coming continuously
    - Loads new data within a minute.
    - If it takes more than a minute, create new smaller files once per minute
    - Best compressed size of file for snowpipe => 100-250MB



- File Size at Stage
  - Parallel loads are limited by number of data files
  - Optimum compressed size of stage file is => 100-250MB
  - Large files at stage should be broken down into smaller files.
    - Example 1 GB can be broken down to 10 x 100MB files
  - In the above scenario automatic parallel load will happen



- Minimize stage file scans
  - Snowflake scans metadata before loading each file.
  - There can be multiple files in stage based on the directory
  - Partitions the stage data into logical paths
    - s3://bucket/united_states/new_york/2022/08/05/01/
  - Concurrent copy jobs can run on each partition
  - Use Purge to remove successfully loaded files

        - Copy into ..purge = TRUE
        - alter table contacts set stage_copy_option = (purge=true)
        - Can be archived
        - Keep checking the redundant loaded files
    - Use pattern to minimize file scan

- Slowest Copy Method
    - copy into contacts from @ext_aws_stage pattern='.*[.]csv' ;

- Flexible Copy Method
    - copy into contacts from @ext_aws_stage/united_states/new_york/2022/08/05/01 pattern='.*[.]csv' ;
    - Works well if files are exported in named order and want to batch load the files

- Fastest Copy Method - Giving the exact file name
    - copy into contacts from @ext_aws_stage/united_states/new_york/2022/08/05/01 file=contacts3.csv;

- Large Files should be compressed
    - Always compress large files
    - If staging uncompressed files, snowflake compress it using gzip
    - csv compressed as gzip gives better performance than parquet / ORC snappy or zippy compression

- Load sorted data to tables
    - Load sorted data
    - It will result in natural clustering. This will avoid cost of reclustering
    - Only applicable for large tables - 1GB and above

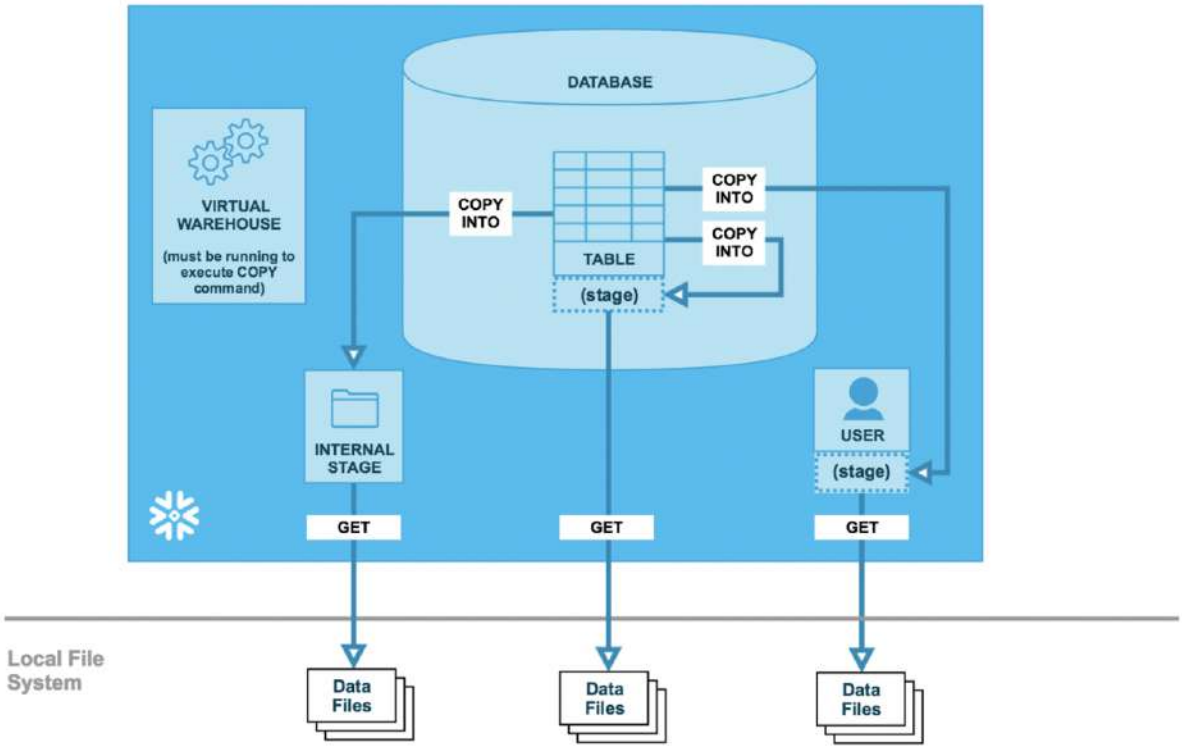
- Load the attributes what is needed
    - Load fields which are needed
    - Not efficient to load all fields
    - Save storage

- External Stage Location
  - If in same availability zone, help to avoid network cost
  - If snowflake account in us-east-2, create external stage in us-east-2

- Separate load and query jobs
  - Dedicated warehouses
  - This will segregate the computing power
  - Homogeneous users on same virtual warehouse

DATA UNLOADING

- COPY INTO <LOCATION>
  - Supports query as well, instead of just Table
  - Supports unload to single or multiple files
- GET



- User Stage

Hands On
-- Create the table for user stage data
create table subjects (name string, sub1 string, sub2 string, sub3 string);

```
-- insert the data
insert into subjects values ('Sharon',  'Economics',    'Art',              'Psychology') ;
insert into subjects values ('Dave',    'Finance',      'Entrepreneurship',   'Personal
Development') ;
insert into subjects values ('Dean',    'Real Estate',  'Digital Product',  'Productivity') ;
insert into subjects values ('Dee',     'Recruitment',  'Staffing',         'Training') ;
insert into subjects values ('Tony',    'Strategy',     'Mentorship',       'Business Mastery') ;

-- check the table
select * from subjects ;

--Copy table into user stage
copy into @~/vsingh from subjects file_format = (type = 'CSV' compression = none);

--Check user stage
list @~/vsingh ;

-- see the data in stage
select t.$1, t.$2, t.$3, t.$4 from @~/vsingh t;
select * from @~/vsingh t; -- doesn't work, works in table stage

-- GET Run via snowsql
-- Check the file at local file system
get @~/vsingh_0_0_0.csv
file:///Users/vipinsingh/Desktop/SnowflakeCourse/Data/UserStageFile/DataUnload;

--Check the file

-- Cleanup
drop table subjects ;
-- remove files for re-run – can’t drop user stage
remove @~/vsingh pattern='.*.csv' ;
list @~/vsingh ;
--delete the file from directory
```

- Table Stage

```
-- Create Table
create table salary (Age number , Sal number) ;
--Insert Data
insert into salary values (50, 10000);
insert into salary values (44, 8000) ;
insert into salary values (61, 10000);
insert into salary values (54, 8000) ;
insert into salary values (45, 8000) ;
-- check the data
select * from salary ;
```

```
--Copy table into user stage
copy into @%salary from salary file_format = (type = 'CSV' compression = none);

--Check table stage
list @%salary;

--see the data
select t.$1, t.$2 from @%salary t;
select * from @%salary; -- works in table stage

-- GET Run via snowsql
-- Check the file at local file system
get @%salary/data_0_0_0.csv
file:///Users/vipinsingh/Desktop/SnowflakeCourse/Data/TableStageFile/DataUnload;
--Rerun overwrites the file
--Check the file in Folder

--cleanup
drop table salary ; -- dropping table will remove the data from stage table_stage
-- OR
remove @%salary pattern='.*.csv' ; -- to reload, we can remove the files.
select * from @%salary t;
```

- Named Internal Stage

```
-- create a table to load internal named stage data
create table weather (city string, temperature number);
--insert
insert into  weather values ('Charlotte',  70) ;
insert into  weather values ('Raleigh',    80) ;
insert into  weather values ('Wilmington', 75) ;
insert into  weather values ('Apex',       73) ;
insert into  weather values ('HollySpring',78) ;

select * from weather ;
-- create internal stage the name is ‘shared_stage’’
create stage shared_stage ;

-- show all stages
show stages ;

-- Describe internal stage
desc stage shared_stage  ;

copy into @shared_stage from weather file_format = (type = 'CSV' compression = none);

--Check table stage
list @shared_stage;
```

```
-- Run it via snowsql, use case for multiple users loading multiple tables.
-- check the file
get @shared_stage/data_0_0_0.csv
file:///Users/vipinsingh/Desktop/SnowflakeCourse/Data/InternalNamedStageFile/DataUnload ;
-- check the file in the local directory

-- Cleanup
drop stage shared_stage ;
drop table weather ;
```

- Unloading using Query

```
-- create a table to load internal named stage data
create table weather (city string, temperature number);
--insert
insert into  weather values ('Charlotte',  70) ;
insert into  weather values ('Raleigh',    80) ;
insert into  weather values ('Wilimngton', 75) ;
insert into  weather values ('Apex',       73) ;
insert into  weather values ('HollySpring',78) ;

select * from weather ;
-- create internal stage the name is ‘shared_stage’’
create stage shared_stage ;

-- show all stages
show stages ;


copy into @shared_stage from (select city from weather) file_format = (type = 'CSV'
compression = none);

--Check table stage
list @shared_stage;


-- Run it via snowsql, use case for multiple users loading multiple tables.
-- check the file
get @shared_stage/data_0_0_0.csv
file:///Users/vipinsingh/Desktop/SnowflakeCourse/Data/InternalNamedStageFile/DataU
nload ;
-- check the file in the local directory

-- Cleanup
drop stage shared_stage ;
drop table weather ;
```

- Unloading to multiple files

```
-- create a table to load internal named stage data
create or replace table items as select * from
SNOWFLAKE_SAMPLE_DATA.TPCDS_SF100TCL.ITEM ;

select * from items ;
-- create internal stage the name is ‘shared_stage’’
create or replace stage shared_stage ;

-- show all stages
show stages ;

remove @shared_stage ;
--file size 16MB is default
copy into @shared_stage from items file_format = (type = 'CSV' compression =
none) ;

copy into @shared_stage from items file_format = (type = 'CSV' compression =
none) single = true max_file_size = 150000000;

copy into @shared_stage from items file_format = (type = 'CSV' compression =
none) max_file_size = 10000000; -- 10MB

copy into @shared_stage from items file_format = (type = 'CSV' compression =
none) max_file_size = 50000000; -- 50MB

--Check table stage
list @shared_stage;

select t.$1, t.$2 from @shared_stage t ;


-- Run it via snowsql, use case for multiple users loading multiple tables.
-- check the file
get @shared_stage/data_0_0_0.csv
file:///Users/vipinsingh/Desktop/SnowflakeCourse/Data/InternalNamedStageFil
e/DataUnload ;
```

```
get @shared_stage/data
file:///Users/vipinsingh/Desktop/SnowflakeCourse/Data/InternalNamedStageFil
e/DataUnload ;
get @shared_stage
file:///Users/vipinsingh/Desktop/SnowflakeCourse/Data/InternalNamedStageFil
e/DataUnload pattern='.*.csv';
-- check the file in the local directory

-- Cleanup
drop stage shared_stage ;
drop table items ;
```

LOADING / UNLOADING - AWS

Storage integration

- Snowflake object
- Stores (IAM) entity for your external cloud storage
- Allowed locations

```
-- Data Loading AWS S3
---------------------------------

-- Create AWS Free Tier Account
-- Create the S3 bucket and the required folders
-- IAM Role for Bucket
-- Create Integration Object - Get the role ARN and copy below in storage_aws_role_arn
--                          copy the S3 URL in storage_allowed_location
-- If You re-create the storage integration object. You need to update the storage_iam_user_arn and storage_aws_external_id
-- at aws role - edit trust relationship
create or replace storage integration int_aws_s3
  type = external_stage
  storage_provider = s3
  enabled = true
  storage_aws_role_arn = 'arn:aws:iam::191937777997:role/snowflake-role-full-access'
                                                    storage_allowed_locations =
('s3://snowflakedatafiles789/csvfiles/dataload/','s3://snowflakedatafiles789/csvfiles/dataunload/')
```

```
  comment = 'This is the integration object for loading the files from AWS S3 to Snowflake' ;

-- Describe Integration Object
desc integration int_aws_s3 ;

-- Copy ARN and External ID in IAM Role at AWS

-- create file format - since it will be used in external stage and copy command. so creating is better
create or replace file format file_format_csv skip_header = 1 compression = none;

--describe file formats
describe file format file_format_csv ;

-- create external stage for loading data
create or replace stage ext_stage_ld
  url = 's3://snowflakedatafiles789/csvfiles/dataload/'
  storage_integration = int_aws_s3
  file_format = file_format_csv ;

-- Show stages
show stages ;

-- list
list @ext_stage_ld ;

  -- Create the table to load
create or replace TABLE WEB_SITES
( WEB_SITE_SK NUMBER(38,0),
  WEB_SITE_ID VARCHAR(16),
  WEB_REC_START_DATE DATE,
  WEB_REC_END_DATE DATE,
  WEB_NAME VARCHAR(50),
  WEB_OPEN_DATE_SK NUMBER(38,0),
  WEB_CLOSE_DATE_SK NUMBER(38,0),
  WEB_CLASS VARCHAR(50),
  WEB_MANAGER VARCHAR(40),
  WEB_MKT_ID NUMBER(38,0),
  WEB_MKT_CLASS VARCHAR(50),
  WEB_MKT_DESC VARCHAR(100),
  WEB_MARKET_MANAGER VARCHAR(40),
  WEB_COMPANY_ID NUMBER(38,0),
  WEB_COMPANY_NAME VARCHAR(50),
  WEB_STREET_NUMBER VARCHAR(10),
  WEB_STREET_NAME VARCHAR(60),
  WEB_STREET_TYPE VARCHAR(15),
  WEB_SUITE_NUMBER VARCHAR(10),
  WEB_CITY VARCHAR(60),
  WEB_COUNTY VARCHAR(30),
  WEB_STATE VARCHAR(2),
```

```
  WEB_ZIP VARCHAR(10),
  WEB_COUNTRY VARCHAR(20),
  WEB_GMT_OFFSET NUMBER(5,2),
  WEB_TAX_PERCENTAGE NUMBER(5,2) );

  -- Copy command
copy into web_sites from @ext_stage_ld ;

  -- select
select * from web_sites ;

  -- Cleanup
drop stage ext_stage_ld ;
drop table web_sites ;



-- Data Unloading -> from snowflake to AWS S3
-----------------------
create table web_pages as select * from
SNOWFLAKE_SAMPLE_DATA.TPCDS_SF100TCL.WEB_PAGE ;

-- Check the data in the table
select * from web_pages ;

 -- create external stage
create or replace stage ext_stage_unld
  url = 's3://snowflakedatafiles789/csvfiles/dataunload/'
  storage_integration = int_aws_s3
  file_format = file_format_csv ;

  -- show stages
show stages ;

  -- list
list @ext_stage_unld ;

  -- Copy for unload
copy into @ext_stage_unld from web_pages ;

  -- Check the data
select t.$1, t.$2, t.$3, t.$4 from @ext_stage_unld t ;

--
remove @ext_stage_unld ;
-- Check the file

  -- cleanup
```

```
drop stage ext_stage_unld ;
drop table web_pages ;
drop storage integration int_aws_s3 ;
drop file format file_format_csv ;
```

LOADING / UNLOADING DATA in AZURE

Handson

```
-- Data Loading Azure
--------------------------------

-- Create Azure Free Tier Account
-- Create the Storage Account under a resource group
-- Create containers and upload file
-- Get the Tenant ID from Active Directory in Azure
-- Create Integration Object - Use the Tenant ID
--                          Click on the consent URL
-- Go to IAM role in Azure and Add Role assignment
          – Go on Storage account and click access control
             --Select a role (Storage Blob Contributor)
          --Assign Access to User, group or service principal
          --Search the Snowflake - It will show the snowflake integration object
          -- Select it.
-- Now the access from Snowflake to Azure containers and Files is done

create or replace storage integration int_azure
  type = external_stage
  storage_provider = azure
  enabled = true
  azure_tenant_id = ''
  storage_allowed_locations
        = ('azure://<storageaccount>.blob.core.windows.net/<container>')
  comment = 'This is the integration object for loading / unloading the files from Azure to Snowflake' ;

-- Describe Integration Object
desc integration int_azure ;


-- create file format - since it will be used in the external stage and copy command, so creating is better
create or replace file format file_format_csv skip_header = 1 compression = none;
```

```
--describe file formats
describe file format file_format_csv ;

-- create external stage for loading data
create or replace stage ext_stage_ld
  url = 'azure://<storageaccount>.blob.core.windows.net/<container>'
  storage_integration = int_azure
  file_format = file_format_csv ;

-- Show stages
show stages ;

-- list
list @ext_stage_ld ;


  -- Create the table to load
create or replace TABLE CALL_CENTERS
( CC_CALL_CENTER_SK NUMBER(38,0), CC_CALL_CENTER_ID VARCHAR(16),
CC_REC_START_DATE DATE, CC_REC_END_DATE DATE,
  CC_CLOSED_DATE_SK NUMBER(38,0), CC_OPEN_DATE_SK NUMBER(38,0), CC_NAME
VARCHAR(50), CC_CLASS VARCHAR(50),
  CC_EMPLOYEES NUMBER(38,0), CC_SQ_FT NUMBER(38,0), CC_HOURS VARCHAR(20),
CC_MANAGER VARCHAR(40), CC_MKT_ID NUMBER(38,0),
  CC_MKT_CLASS VARCHAR(50), CC_MKT_DESC VARCHAR(100), CC_MARKET_MANAGER
VARCHAR(40), CC_DIVISION NUMBER(38,0),
  CC_DIVISION_NAME VARCHAR(50), CC_COMPANY NUMBER(38,0), CC_COMPANY_NAME
VARCHAR(50), CC_STREET_NUMBER VARCHAR(10),
  CC_STREET_NAME VARCHAR(60), CC_STREET_TYPE VARCHAR(15), CC_SUITE_NUMBER
VARCHAR(10), CC_CITY VARCHAR(60),
  CC_COUNTY VARCHAR(30), CC_STATE VARCHAR(2), CC_ZIP VARCHAR(10), CC_COUNTRY
VARCHAR(20), CC_GMT_OFFSET NUMBER(5,2),
  CC_TAX_PERCENTAGE NUMBER(5,2)
);

select * from call_centers ;

  -- Copy command
copy into call_centers from @ext_stage_ld ;

  -- select
select * from call_centers ;

  -- Cleanup

drop table call_centers ;
drop stage ext_stage_ld ;
```

```
-- Data Unloading -> from snowflake to Azure
-----------------------
create table household_demographics as select * from
SNOWFLAKE_SAMPLE_DATA.TPCDS_SF100TCL.household_demographics ;

-- Check the data in the table
select * from household_demographics ;

select count(*) from household_demographics ;

 -- create external stage
create or replace stage ext_stage_unld
  url = 'azure://<storageaccount>.blob.core.windows.net/<container>'
  storage_integration = int_azure
  file_format = file_format_csv ;

  -- show stages
show stages ;

  -- list
list @ext_stage_unld ;

  -- Copy for unload
copy into @ext_stage_unld from household_demographics ;

  -- Check the data
select t.$1, t.$2, t.$3, t.$4 from @ext_stage_unld t ;

--
remove @ext_stage_unld ;
-- Check the file

  -- cleanup

drop stage ext_stage_unld ;
drop table household_demographics ;
drop storage integration int_azure ;
drop file format file_format_csv ;
```

LOADING / UNLOADING DATA in GCP

```
-- Data Loading GCP
--------------------------------

-- Create GCP Free Tier Account
-- Create the Bucket under Cloud storage
-- Create Folders and Upload a file
-- Create Integration Object
   -- Copy STORAGE_GCP_SERVICE_ACCOUNT property value
-- Go to Bucket in GCP
   -- Click on VIEW INFO FIELD
   -- Add members (Paste the Property Value)
   -- Select a Role
      --Cloud Storage - Administrator


create or replace storage integration int_gcp
  type = external_stage
  storage_provider = gcs
  enabled = true
  storage_allowed_locations
        = ('gcs://<bucket>/folder')
  comment = 'This is the integration object for loading / unloading the files from GCP to Snowflake' ;

-- Describe Intergation Object
desc integration int_gcp ;









-- create file format - since it will be used in external stage and copy command. so creating is better
create or replace file format file_format_csv skip_header = 1 compression = none;

--describe file formats
describe file format file_format_csv ;

-- create external stage for loading data
create or replace stage ext_stage_ld
```

```
  url = 'gcs://<bucket>/<folder>'
  storage_integration = int_gcp
  file_format = file_format_csv ;

-- Show stages
show stages ;

-- list
list @ext_stage_ld ;

  -- Create the table to load
create or replace TABLE PROMOTIONS
( P_PROMO_SK NUMBER(38,0), P_PROMO_ID VARCHAR(16), P_START_DATE_SK
NUMBER(38,0), P_END_DATE_SK NUMBER(38,0),
  P_ITEM_SK NUMBER(38,0), P_COST NUMBER(15,2), P_RESPONSE_TARGET NUMBER(38,0),
P_PROMO_NAME VARCHAR(50),
  P_CHANNEL_DMAIL VARCHAR(1), P_CHANNEL_EMAIL VARCHAR(1), P_CHANNEL_CATALOG
VARCHAR(1), P_CHANNEL_TV VARCHAR(1),
  P_CHANNEL_RADIO VARCHAR(1), P_CHANNEL_PRESS VARCHAR(1), P_CHANNEL_EVENT
VARCHAR(1), P_CHANNEL_DEMO VARCHAR(1),
  P_CHANNEL_DETAILS VARCHAR(100), P_PURPOSE VARCHAR(15), P_DISCOUNT_ACTIVE
VARCHAR(1)
);

select * from promotions ;

  -- Copy command
copy into promotions from @ext_stage_ld ;

  -- select
select * from promotions ;

  -- Cleanup

drop table promotions ;
drop stage ext_stage_ld ;
```

```
-- Data Unloading -> from snowflake to GCP
-----------------------
create table stores as select * from SNOWFLAKE_SAMPLE_DATA.TPCDS_SF100TCL.store ;

-- Check the data in the table
select * from stores ;
```

```
select count(*) from stores ;

 -- create external stage
create or replace stage ext_stage_unld
  url = 'gcs://<bucket>'
  storage_integration = int_gcp
  file_format = file_format_csv ;

  -- show stages
show stages ;

  -- list
list @ext_stage_unld ;

  -- Copy for unload
copy into @ext_stage_unld from stores ;

  -- Check the data
select t.$1, t.$2, t.$3, t.$4 from @ext_stage_unld t ;

--
remove @ext_stage_unld ;
-- Check the file

  -- cleanup

drop stage ext_stage_unld ;
drop table stores ;
drop storage integration int_gcp ;
drop file format file_format_csv ;
```

LOADING SEMI- STRUCTURED DATA

Semi-structured data

- Data comes from multiple sources like sensors, applications in semi-structured formats
- Data is nested and hierarchical in nature.
- Snowflake provide built in support for following semi-structured formats
    - JSON
        - ➢ Java Script Notation
        - ➢ Plain text
    - Avro
        - ➢ Framework developed for use with Apache hadoop
    - ORC
        - ➢ Optimized Row Columnar
        - ➢ Binary format used to store hive data
    - Parquet
        - ➢ Columnar data designed for Hadoop Systems
    - XML
        - ➢ Commonly used to interchange data on web

- Data Types to Support Semi-structured data
    - VARIANT
        - ➢ 16MB, can contain any data type
        - ➢ to_variant, expression::variant, :: -> Cast Operator
        - ➢ Used to create hierarchical data
    - OBJECT
        - ➢ Key, Value Pairs
        - ➢ Key- VARCHAR, Value - VARIANT
        - ➢ Delimited by curly braces { }
        - ➢ Value retrieved by square bracket or colon
        - ➢ {}, {Key1:Value1, Key2:Value2} -> {}
        - ➢ Object_construct
    - ARRAY
        - ➢ 0 or more values, referred by
        - ➢ Element accessed by position
        - ➢ Data Type of element is VARIANT
        - ➢ Can grow dynamically, ARRAY_APPEND
        - ➢ Delimited by square bracket
        - ➢ [], [‘Washington’,’Raleigh’]
        - ➢ array_construct

```
-- Plain text data
select 'book_name','Snowflake Fundamentals','pages',100,'author','John' ;
select 'Data Science Basics','Snowflake Fundamentals','Oracle Database Development','Basics of machine learning' ;


-- constructs
select object_construct ('book_name','Snowflake Fundamentals','pages',100,'author','John') as object_book ;
select array_construct ('Data Science Basics','Snowflake Fundamentals','Oracle Database Development','Basics of machine learning') as array_books ;

-- create tables
create table book (book_info) as select object_construct ('book_name','Snowflake Fundamentals','pages',100,'author','John') as object_book;
drop table book ;
select * from book ;
select book_info:author as book_author, book_info:pages as no_of_pages from book ;

select book_info:author::string as book_author, book_info:pages as no_of_pages from book ;


-- Array

create table books (book_arr) as select array_construct ('Data Science Basics','Snowflake Fundamentals','Oracle Database Development','Basics of machine learning') as array_books ;

select * from books ;

select book_arr[0],book_arr[1],book_arr[2],book_arr[3] from books ;

select book_arr[0]::string as book1,book_arr[1]::string as book2,
      book_arr[2]::string as book3,book_arr[3]::string as book4
from books ;

--
-- Entire JSON/AVRO/ORC/PARQUET files are stored in VARIANT

-- clean up

drop table book ;
drop table books ;
```

So far we have seen.

1. Upload File at External or Internal Location
2. Create File Format
3. Create Internal or External named Stage
4. COPY in to <table> -> LOAD

   OR

   COPY into <location> -> UNLOAD

When we deal with Semi-Structured data we have to do the following

1. Directly insert the data into Table if available as text and small

   OR

1. Upload File at External or Internal Location
2. Create File Format
3. Create External named Stage
4. COPY in to <table> - Raw Data => VARIANT data type
5. Parse
6. Flatten
7. Load

In Snowflake We favor ELT over ETL Load First and Transform later..

In Order to understand further, Lets go though our Demo file

Demo File: book.csv, books.csv

Load the JSON Data

```
-- Create table to load the JSON data
create or replace table raw_books (src variant);

-- Check the data
select * from raw_books ;

-- Insert data using parse_json

insert into raw_books
select parse_json(column1) as src from values
('{
        "name": "Snowflake Fundamentals",
        "author_first_name":"John",
        "author_last_name":"Doe",
        "pages":250,
        "publication_years":[2012, 2014, 2015, 2018],
        "languages":["English","French","German","Hindi"],
        "publisher":
                {"name":"Highland Publication Limited",
                 "street":"2434 Senora Lane",
                 "city":"Richmond",
                 "state":"Virginia",
```

```
                    "country":"US"
                    },
          "contents":[
                    {"section":"Snowflake Architecture",
"chapters":["Introduction","Architecture","Performance management"]},
                    {"section":"Access Management and Snowflake Objects",
"chapters":["Roles","Hierarchy","Tables and Views","Stored Procedures"]},
                    {"section":"Data Movement 1", "chapters":["Data Loading","Data Unloading"]},
                    {"section":"Data Movement 2", "chapters":["Semi-Structured Data","SnowPipe"]},
                    {"section":"Dynamic Data Masking", "chapters":["Introduction","Best Practices"]},
                    {"section":"Data Sharing and Data Protection", "chapters":["Zero Copy
Cloning","Tasks","Streams"]}
          ]
}
 ') ;

-- Check the data
select * from raw_books ;

-- truncate table and insert by COPY command
truncate table raw_books ;

-- Create integration object
create or replace storage integration int_gcp
  type = external_stage
  storage_provider = gcs
  enabled = true
  storage_allowed_locations
        = ('gcs://snowflakedatafiles789/jsonfiles/dataload')
  comment = 'This is the integration object for loading / unloading the files from GCP to Snowflake' ;

-- Describe
desc integration int_gcp ;


-- create file format - since it will be used in external stage and copy command. so creating is better
create or replace file format file_format_json type = JSON ;

--describe file formats
describe file format file_format_json ;

-- create external stage for loading data
create or replace stage ext_stage_ld
  url = 'gcs://snowflakedatafiles789/jsonfiles/dataload/'
  storage_integration = int_gcp
  file_format = file_format_json ;

-- Show stages
show stages ;
```

```
-- list
list @ext_stage_ld ;

  -- Copy command
copy into raw_books from @ext_stage_ld files = ('book.json');

  -- select
select * from raw_books ;

-- Load the books.json
truncate table raw_books ;
copy into raw_books from @ext_stage_ld files = ('books.json');

-- Check the data
select * from raw_books ;


-- Parsing the JSON data
-- select specific columns
select src:name as name, src:author_first_name as author_first_name, src:author_last_name as author_last_name, src:pages as pages from raw_books ;

-- type case as string and int
select src:name::string as name, src:author_first_name::string as author_first_name,
       src:author_last_name::string as author_last_name, src:pages::int as pages
 from raw_books ;

 -- Handling Array - selecting the string array and number array
   select src:name::string as name, src:publication_years as publication_years, src:languages as languages from raw_books ;

 -- select first element of Array
 select src:name::string as name, src:publication_years[0] as publication_years, src:languages[0]::string as languages from raw_books ;

 -- ARRAY_SIZE - to check the length of array and do operations traversing in loop etc
    select array_size(src:publication_years) as pub_year_arr_size, array_size(src:languages) as language_arr_size from raw_books ;


Nested object

select
       rb.src:publisher.name::string as publisher_name,
       rb.src:publisher.street::string as publisher_street,
       rb.src:publisher.city::string as publisher_city,
       rb.src:publisher.state::string as publisher_state,
       rb.src:publisher.country::string as publisher_country
```

```
from raw_books rb ;

  select * from raw_books ;

 -- FLATTEN (explodes compound into multiple)
    -- table function which takes variant, object and array column and produces a lateral view
    -- used for converting semi-structured to relational
      -- input is expr -> object, array or variant. This will be converted to rows and will be a mandatory parameter
    -- output is - (seq, key, path, index,value,this)
     -- (generated number, key of exploded expression, path of the element, index if array, value, element )
--
  select src:name::string as name, src:publication_years as publication_years, src:languages as languages from raw_books ;

-- Lets see the example of publication_years
select * from raw_books rb, lateral flatten(input=> rb.src:publication_years) py where src:name = 'Snowflake Fundamentals' ;

-- object
select * from raw_books rb, lateral flatten(input=> rb.src:publisher) py where src:name = 'Snowflake Fundamentals' ;

-- variant
select * from raw_books rb, lateral flatten(input=> rb.src) py where src:name = 'Snowflake Fundamentals' ;



-- Lets start on the data

-- rb.src:name::string as name,
select rb.src:name::string as name, src:publication_years as publication_years, py.index, py.value
 from raw_books rb, lateral flatten(input=> rb.src:publication_years) py ;

 -- You can use as table as well
 select rb.src:name::string as name, src:publication_years as publication_years, py.index, py.value
 from raw_books rb, table(flatten(rb.src:publication_years)) py ;

 ---
 -- Lets see the example of languages
select * from raw_books rb, lateral flatten(input=> rb.src:languages) ;

-- rb.src:name::string as name,
select rb.src:name::string as name, src:publication_years as publication_years, lng.index, lng.value::string as value
 from raw_books rb, lateral flatten(input=> rb.src:languages) lng ;
```

```
 -- You can use as table as well
   select rb.src:name::string as name, src:publication_years as publication_years, lng.index,
lng.value::string as value
 from raw_books rb, lateral flatten(input=> rb.src:languages)) lng ;



--Final Query
select --element
       rb.src:name::string as name,
       rb.src:author_first_name::string as author_first_name,
       rb.src:author_last_name::string  as author_last_name,
       rb.src:pages as pages,
       --Array
       pyr.value::string as publication_year,
       --Array
       lng.value::string as language,
       --nested object
       rb.src:publisher.name::string as publisher_name,
       rb.src:publisher.street::string as publisher_street,
       rb.src:publisher.city::string as publisher_city,
       rb.src:publisher.state::string as publisher_state,
       rb.src:publisher.country::string as publisher_country,
       -- nested array of objects. Object also has an array -- more complicated version
       cnt.value:section::string as section,
       chp.value::string as chapters
from raw_books rb,
     table(flatten(rb.src:publication_years)) pyr,
     table(flatten(rb.src:languages)) lng,
     table(flatten(rb.src:contents)) cnt,
     table(flatten(cnt.value:chapters)) chp
 where src:name = 'Snowflake Fundamentals';
-- where src:name = 'Snowflake Advanced - Architect' ;




--- Create the final table from JSON file
create or replace table parsed_books as
select --element
       rb.src:name::string as name,
       rb.src:author_first_name::string as author_first_name,
       rb.src:author_last_name::string  as author_last_name,
       rb.src:pages as pages,
       --Array
```

```
    pyr.value::string as publication_year,
    --Array
    lng.value::string as language,
    --nested object
    rb.src:publisher.name::string as publisher_name,
    rb.src:publisher.street::string as publisher_street,
    rb.src:publisher.city::string as publisher_city,
    rb.src:publisher.state::string as publisher_state,
    rb.src:publisher.country::string as publisher_country,
    -- nested array of objects. Object also has an array -- more complicated version
    cnt.value:section::string as section,
    chp.value::string as chapters
from raw_books rb,
   table(flatten(rb.src:publication_years)) pyr,
   table(flatten(rb.src:languages)) lng,
   table(flatten(rb.src:contents)) cnt,
   table(flatten(cnt.value:chapters)) chp;

-- select the relational form of JSON data
select * from parsed_books where name = 'Snowflake Fundamentals' and language = 'German' and publication_year = 2018;
select * from parsed_books where name = 'Snowflake Advanced - Administrator' and language = 'English' and publication_year = 2020;
select * from parsed_books where name = 'Snowflake Advanced - Architect' and language = 'English' and publication_year = 2020;
select * from parsed_books where name = 'Snowflake Advanced - Data Engineer' and language = 'English' and publication_year = 2021;
select * from parsed_books where name = 'Snowflake Advanced - Data Scientist' and language = 'English' and publication_year = 2022;

 -- cleanup
drop table raw_books ;
drop table parsed_books ;
drop stage ext_stage_ld ;
drop storage integration int_gcp ;
drop file format file_format_json ;
```

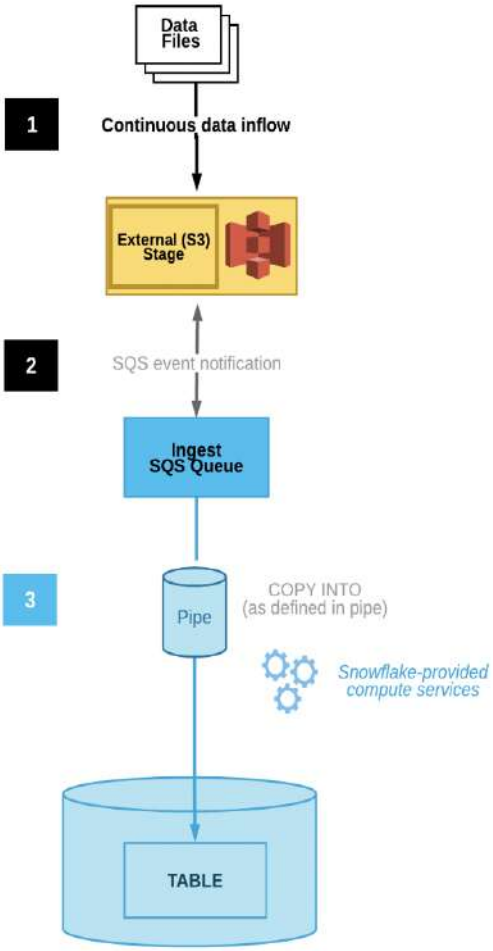
```
-- PARQUET DATA LOAD

-- Data - sample one record

-- {
--   "continent": "Europe",
--   "country": {
--     "city": [
--       "Paris",
--       "Nice",
--       "Marseilles",
--       "Cannes"
--     ],
--     "name": "France"
--   }
-- }

--create table to load the above parquet file
create or replace table raw_cities (src variant);

-- Check the data
select * from raw_cities ;


-- Create integration object
create or replace storage integration int_gcp
  type = external_stage
  storage_provider = gcs
  enabled = true
  storage_allowed_locations
        = ('gcs://snowflakedatafiles789/parquetfiles/dataload')
  comment = 'This is the integration object for loading / unloading the files from GCP to Snowflake' ;

-- Describe
desc integration int_gcp ;


-- create file format - since it will be used in external stage and copy command. so creating is better
create or replace file format file_format_parquet type = PARQUET ;

--describe file formats
describe file format file_format_parquet ;

-- create external stage for loading data
create or replace stage ext_stage_ld
  url = 'gcs://snowflakedatafiles789/parquetfiles/dataload/'
  storage_integration = int_gcp
  file_format = file_format_parquet ;
```

```
-- Show stages
show stages ;

-- list
list @ext_stage_ld ;

-- check stage
select * from @ext_stage_ld ;
--

  -- Copy command
copy into raw_cities from @ext_stage_ld
files = ('cities.parquet');

-- Parsing the CITIES data

--PARQUET DATA PARSING

select * from raw_cities  ;


 -- Handling Array - selecting the arrays
  select src:continent::string, src:country:name::string as country_name, src:country:city as cities  from raw_cities ;


 -- FLATTEN (explodes compound into multiple)
    -- table function which takes variant, object and arrat column and produces a lateral view
    -- used for converting semi-structured to relational
      -- input is expr -> object, array or variant. This will be converted to rows and will be a mandatory parameter
    -- output is - (seq, key, path, index,value,this)
     -- (generated number, key of exploded expression, path of the element, index if array, value, element )

-- Lets see the example
select rct.src:continent::string, rct.src:country:name::string as country_name, r.value::string as cities
from raw_cities rct,  lateral flatten(input=> rct.src:country:city) r;


--- Create the final table from PARQUET file
create or replace table parsed_cities as
select rct.src:continent::string as continent, rct.src:country:name::string as country_name, r.value::string as cities
from raw_cities rct,  lateral flatten(input=> rct.src:country:city) r;

-- select the relational form of PARQUET data
select * from parsed_cities where continent = 'Europe' and country_name = 'Greece' ;
select * from parsed_cities where continent = 'Europe' and country_name = 'France' ;
```

```
select * from parsed_cities where continent = 'North America' and country_name = 'Canada' ;


  -- cleanup
drop table raw_cities ;
drop table parsed_cities ;
drop stage ext_stage_ld ;
drop storage integration int_gcp ;
drop file format file_format_parquet ;
```

SNOWPIPE

- If the files are coming continuously
- Loads new data within a minute. Compute managed by Snowflake
- If it takes more than a minute, create new smaller files once per minute
- Best compressed size of file for snowpipe => 100-250MB



Next Steps

- Use the existing S3 bucket
- Create the integration object
- Create file format
- Create external stage
- Create the table to load the data via pipe
- Create and test the copy command
- Create Pipe
- Create S3 notification
- Place the files in S3 bucket and validate the load

```
create or replace storage integration int_aws_s3
  type = external_stage
  storage_provider = s3
  enabled = true
  storage_aws_role_arn = 'arn:aws:iam::191937777997:role/snowflake-role-full-access'
  storage_allowed_locations = ('s3://snowflakedatafiles789/csvfiles/dataload/','s3://snowflakedatafiles789/csvfiles/dataunload/')
  comment = 'This is the integration object for loading the files from AWS S3 to Snowflake' ;

-- Describe Intergation Object
desc integration int_aws_s3 ;

-- Copy ARN and External ID in IAM Role at AWS

-- create file format - since it will be used in the external stage and copy command. so creating is better
create or replace file format file_format_csv compression = none;

--describe file formats
describe file format file_format_csv ;

-- create a snowpipe folder at s3 bucket and add a file promotions1.csv

-- create external stage for loading data
create or replace stage ext_stage_ld
  url = 's3://snowflakedatafiles789/csvfiles/dataload/snowpipe'
  storage_integration = int_aws_s3
  file_format = file_format_csv ;

-- if error
-- copy storage_aws_external_id in the snowflake-full-access role trust relationship

-- Show stages
show stages ;

-- list
```

```
list @ext_stage_ld ;

  -- Create the table to load
create or replace TABLE PROMOTIONS
(     P_PROMO_SK     NUMBER(38,0),     P_PROMO_ID     VARCHAR(16),     P_START_DATE_SK
NUMBER(38,0), P_END_DATE_SK NUMBER(38,0),
   P_ITEM_SK NUMBER(38,0), P_COST NUMBER(15,2), P_RESPONSE_TARGET NUMBER(38,0),
P_PROMO_NAME VARCHAR(50),
   P_CHANNEL_DMAIL VARCHAR(1), P_CHANNEL_EMAIL VARCHAR(1), P_CHANNEL_CATALOG
VARCHAR(1), P_CHANNEL_TV VARCHAR(1),
    P_CHANNEL_RADIO VARCHAR(1), P_CHANNEL_PRESS VARCHAR(1), P_CHANNEL_EVENT
VARCHAR(1), P_CHANNEL_DEMO VARCHAR(1),
    P_CHANNEL_DETAILS VARCHAR(100), P_PURPOSE VARCHAR(15), P_DISCOUNT_ACTIVE
VARCHAR(1)
);

select * from promotions ;

  -- Copy command working
copy into promotions from @ext_stage_ld ;

  -- select
select * from promotions ;

-- truncate table and remove file promotions1.csv from S3

truncate table promotions ;

--create pipe
create or replace pipe pipe_for_promotions
auto_ingest = TRUE
as
copy into promotions from @ext_stage_ld ;

--describe pipe
describe pipe pipe_for_promotions ;

-- copy notification_channel value
   -- Go to AWS console
   -- Click on Bucket
   -- Click on Properties
   -- Scroll Down to Create Event Notification
   -- click on create event notification
   -- Give event name and prefix
   -- Event Types - All object create events
   -- Go to Destination
   -- select SQS Queue
   -- Paste the notification_channel value under "Enter SQS queue ARN"
   -- Click on Save
```

```
-- Remove file promotion1.csv from snowpipe

-- Keep Adding files to snowpipe folder and
-- Check the data
select * from promotions ;

-- Add the promotions1.csv to snowpipe folder in the bucket


-- Check the status of pipe
select system$pipe_status('pipe_for_promotions') ;

-- Refresh the pipe
alter pipe pipe_for_promotions refresh ;

-- Error message
select * from table(validate_pipe_load(pipe_name => 'pipe_for_promotions',
start_time=>dateadd(hour,-5,current_timestamp()))) ;

-- Error in History
select * from table(
                    information_schema.copy_history(table_name => 'promotions', start_time=>
dateadd(hour,-5,current_timestamp())                    )
   );




-- Troubleshooting the Pipe
-- Check the status of pipe
select system$pipe_status('pipe_for_promotions') ;
-- Introduce an error in the promotions4.csv
-- Create a file promotions4error.csv

select * from promotions order by 1 desc;

-- Error message more generic
select * from table(validate_pipe_load(pipe_name => 'pipe_for_promotions',
start_time=>dateadd(hour,-2,current_timestamp()))) ;

-- Error in History - more specific
select * from table(
                    information_schema.copy_history(table_name => 'promotions', start_time=>
dateadd(hour,-2,current_timestamp())))
   order by 3 desc;
```

```
-- Lets try to refresh and see, if it loads the file once we fixed the error
alter pipe pipe_for_promotions refresh ;


-- Two ways to load the files --
 --    1. Manually by copy command OR

 copy into promotions from @ext_stage_ld files = ('/promotions4err.csv');

 select * from promotions order by 1 desc;
 --OR  2 Rename the File and replace it. Place the New file and wait for Pipe to load
 delete from promotions where p_promo_sk between 1501 and 2000 ;
 -- check pipe

 -- Now remove the error File .
 -- Demonstrate how the Pipe refresh will work, if files are already available to load one
 -- OR 3 truncate the table and recreate the pipe again and add the errored file again
 -- and refresh it

 -- Recreating Pipe steps
 desc pipe pipe_for_promotions ;
                                                                                  --
arn:aws:sqs:us-east-2:079828672386:sf-snowpipe-AIDARFFRI36BLT5VFBWHR-PvH1rcdaWXkPWUD
eVlu4sw

-- pause, check status, recreate, pause, check config, resume pipe, check status
   truncate table promotions ;

   alter pipe pipe_for_promotions set pipe_execution_paused = true ;
   select system$pipe_status('pipe_for_promotions') ;
   -- Recreate Pipe
   create or replace pipe pipe_for_promotions auto_ingest = TRUE as
   copy into promotions from @ext_stage_ld;
   desc pipe pipe_for_promotions ;

   select system$pipe_status('pipe_for_promotions') ;

   select * from promotions order by 1 desc ;

   alter pipe pipe_for_promotions refresh ;

      -- Error message more generic
              select * from table(validate_pipe_load(pipe_name => 'pipe_for_promotions',
start_time=>dateadd(hour,-2,current_timestamp()))) ;

-- Error in History - more specific
  select * from table(
```

```
information_schema.copy_history(table_name => 'promotions', start_time=>
dateadd(hour,-1,current_timestamp())))
  order by 3 desc;

  -- Now place the promotion5.csv and see that pipe is working correctly

  -- Summary, an errored file shows error correctly.
  -- Refresh doesnot load it
  -- you can either manually load it or rename a file and load it

  -- Refresh functionality how it works for Pipe..by adding an extra file.

  ----


-- More on Pipes
desc pipe pipe_for_promotions ;

show pipes ;

show pipes like '%promotions%' ;

show pipes in database training ;

show pipes in schema demo;

show pipes like '%promotion%' in schema demo ;


-- Pipes Do not support PURGE COPY Option
      -- You need use remove ..to delete the staged files
show stages ;

list @ext_stage_ld ;

remove @ext_stage_ld/promotions5.csv ;
delete from promotions where p_promo_sk between 2001 and 2500 ;


-- How to change the referenced stage url, storage integration and encryption
-- Pause, modify the stage, resume pipe
  alter pipe pipe_for_promotions set pipe_execution_paused = true ;
  select system$pipe_status('pipe_for_promotions') ;

  alter stage ext_stage_ld
  set url = 's3://snowflakedatafiles789/csvfiles/dataload/snowpipe/file5' ;

  list @ext_stage_ld ;
```

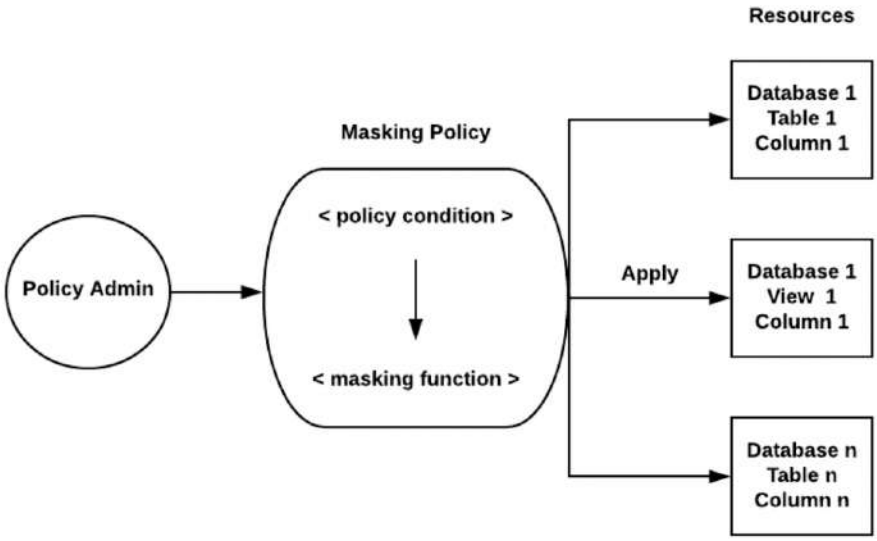
```
  alter pipe pipe_for_promotions set pipe_execution_paused = false ;
  select system$pipe_status('pipe_for_promotions') ;

  -- Hence You need to recreate it,
    -- Pause, modify the stage, recreate pipe,  check status
  create or replace pipe pipe_for_promotions auto_ingest = TRUE as
  copy into promotions from @ext_stage_ld;
  select system$pipe_status('pipe_for_promotions') ;
  -- upload promotions5 in file5 folder

  -- check the output
  select * from promotions order by 1 desc;


-- how can you change pipe. You can not alter the pipe, you need to recreate
-- for example if the table is renamed, and there is an existing pipe.
-- So we need to recreate the pipe.
alter table promotions rename to raw_promotions ;
-- truncate the table
truncate table raw_promotions ;
-- First pause the pipe
alter pipe pipe_for_promotions set pipe_execution_paused = true ;
--check the status of pipe if it is paused
select system$pipe_status('pipe_for_promotions');
--create the pipe again
create or replace pipe pipe_for_promotions
auto_ingest = true
as
copy into raw_promotions from @ext_stage_ld ;
--check the pipe for files sent
alter pipe pipe_for_promotions refresh ;
-- The metadata says that the files are already copied
select system$pipe_status('pipe_for_promotions') ;

-- Data is successfully loaded via refresh
select * from raw_promotions ;

show stages ;



-- cleanup
-- remove folder snowpipe ;
-- remove files from snowpipe ;
-- delete event in AWS console ;
drop integration int_aws_s3 ;
drop table raw_promotions ;
drop file format file_format_csv ;
drop stage ext_stage_ld ;
```

```
drop pipe pipe_for_promotions ;
```

DYNAMIC DATA MASKING

Column Level Security include two features

a) Dynamic Data masking
b) External Tokenization - uses external functions to apply data masking at account level
    i) External functions are user defined functions stored and executed external to snowflake
    ii) This feature is useful to be used on data stored externally.



Authorized role (i.e. SUPPORT)

ID	Phone	SSN
101	408-123-5534	387-78-3456
102	510-334-3564	226-44-8908
103	214-553-9787	359-9987-0098

Unauthorized role (i.e. ANALYST)

ID	Phone	SSN
101	***-**-5534	********
102	***-**-3564	********
103	***-**-9787	********

Demo:

```
-- Dynamic Data Masking Overview and Creation

-- Create a table to demonstrate the Data masking
create or replace table client
(  id NUMBER(38,0), first_name VARCHAR(16), last_name VARCHAR(50),
   sex VARCHAR(1), ethinicity VARCHAR(30), ssn VARCHAR(15),
   street_address VARCHAR(90),status VARCHAR(10)
);

-- insert some data
delete from client ;
insert into client values (111111, 'James', 'Schwartz', 'M', 'American','342-76-9087','5676 Washington Street','ACTIVE') ;
insert into client values (222222, 'Jessica', 'Escobar', 'F', 'Hispanic','456-93-5629','3234 WateringCan Drive','INACTIVE') ;
insert into client values (333333, 'Ben', 'Hardy', 'M', 'American','876-98-3245','6578 Historic Circle','INACTIVE') ;
insert into client values (444444, 'Anjali', 'Singh', 'F', 'Indian American','435-87-6532','8978 Autumn Day Drive','ACTIVE') ;
insert into client values (555555, 'Dean', 'Tracy', 'M', 'African','767-34-7656','2343 India Street','ACTIVE') ;

select * from client ;
```

```
-- Create a View also to demonstrate the data masking
create or replace view vw_client as select * from client where sex = 'M' ;
select * from vw_client ;

-- We will show how can we put masking policies on the table and view columns

-- create roles
create or replace role support ;
create or replace role analyst ;

-- assigning table select  grants to role
grant select on table client to role support;
grant select on table client to role analyst;


-- grant schema usage to roles
grant usage on database training to role support;
grant usage on database training to role analyst;

-- grant schema usage to roles
grant usage on schema demo to role support;
grant usage on schema demo to role analyst;

-- grant usage on warehouse to roles
grant usage on warehouse compute_wh to role support;
grant usage on warehouse compute_wh to role analyst;

-- Now user should have these roles
grant role analyst to user vksingh ;
grant role support to user vksingh ;

-- grant select on view vw_client to roles
grant select on view vw_client to role support;
grant select on view vw_client to role analyst;

-- create a simple masking policy
create or replace masking policy pcy_text as (val varchar) returns varchar -> case when current_role()
in ('ANALYST') then '********' else val end ;

show masking policies ;

-- Since the policy is created, Let's apply it on view and table column

-- Apply it on a table column
alter table client modify column first_name set masking policy pcy_text ;

-- check the table
select * from client ;
```

```
-- Set the role as ANALYST
use role analyst ;
use warehouse compute_wh ;
use database training ;
use schema demo ;

select * from client ;

-- Set the role as support
use role support ;
use warehouse compute_wh ;
use database training ;
use schema demo ;

select * from client ;

-- Set roles to apply policy
use role accountadmin ;
use warehouse compute_wh ;
use database training ;
use schema demo ;

-- Apply it on a view column
alter view vw_client modify column first_name set masking policy pcy_text ;

-- Set role to see data
use role analyst ;
use warehouse compute_wh ;
use database training ;
use schema demo ;
-- check view data
select * from vw_client ;
-- set support role
use role support ;
use warehouse compute_wh ;
use database training ;
use schema demo ;
-- check view data
select * from vw_client ;
```

```
-- Modify Masking Policies

-- So far we have applied the policies it is a good idea to see more properties of masking policies
-- Set roles to account admin
use role accountadmin ;
use warehouse compute_wh ;
use database training ;
use schema demo ;

show masking policies ;
desc masking policy pcy_text ;


-- How can we alter making policies
alter masking policy pcy_text set comment = 'This is a test masking policy on client table and vw_client
view' ;

show masking policies ;
-- works as expected post alter
use role analyst ;
use warehouse compute_wh ;
use database training ;
use schema demo ;
-- check view data
select * from client ;
select * from vw_client ; -- view

-----
use role accountadmin ;
use warehouse compute_wh ;
use database training ;
use schema demo ;

desc masking policy pcy_text ;
-- change it
alter masking policy pcy_text set body -> case when current_role() in ('ANALYST') then '#######' else
val end ;
--
use role analyst ;
use warehouse compute_wh ;
use database training ;
use schema demo ;
-- check view data
select * from client ;
select * from vw_client ; -- view

-- rename it

use role accountadmin ;
```

```
use warehouse compute_wh ;
use database training ;
use schema demo ;

alter masking policy pcy_text rename to pcy_text1;

show masking policies ;
desc masking policy pcy_text ;


use role analyst ;
use warehouse compute_wh ;
use database training ;
use schema demo ;
-- check view data
select * from client ; -- table
select * from vw_client ; -- view

-- How to check which columns and table the masking policy is applied to
use role accountadmin ;
use warehouse compute_wh ;
use database training ;
use schema demo ;

select * from table(information_schema.policy_references(policy_name=>'pcy_text')) ;
select * from table(information_schema.policy_references(policy_name=>'pcy_text1')) ;
```

```
-- Drop Masking Policies, Multiple columns policies and Nested policies

-- How to drop a masking policy, OR recreate it.
drop masking policy pcy_text1 ;  -- gives error and can not be dropped

-- Hence We need to unset it first, we check the objects on which the policy is currently apploed
select * from table(information_schema.policy_references(policy_name=>'pcy_text1')) ;
-- unset the policy
alter table client modify column first_name unset masking policy ;
alter view  vw_client modify column first_name unset masking policy ;
-- check again
```

```
select * from table(information_schema.policy_references(policy_name=>'pcy_text1')) ; -- no results
--show policies ;
show masking policies ;

-- Now we can drop the policy
drop masking policy pcy_text1 ; -- dropped successfully.

-- Now we can recreate the policy
create or replace masking policy pcy_text as (val varchar) returns varchar -> case when current_role()
in ('ANALYST') then '$$$$$$$' else val end ;
--apply it , can apply on multiple columns as well..
select * from client ;

alter table client modify column last_name  set masking policy pcy_text ;
alter table client modify column ethinicity set masking policy pcy_text ;
alter table client modify column status set masking policy pcy_text ;
alter table client modify column ssn set masking policy pcy_text ;


select * from vw_client ;

alter view vw_client modify column first_name set masking policy pcy_text ;
alter view vw_client modify column street_address set masking policy pcy_text ;
--check
select * from table(information_schema.policy_references(policy_name=>'pcy_text')) ;

-- Validate the data in view and table
use role analyst ;
use warehouse compute_wh ;
use database training ;
use schema demo ;
-- check view data
select * from client ; -- table
select * from vw_client ; -- view - so what we have observed here is - If the table has a masking policy
and there is a view..masking gets inherited.
-- We see 6 columns of view masked, but we applied masking on two columns.
use role support ;
use warehouse compute_wh ;
use database training ;
use schema demo ;
-- check view data
select * from client ; -- table
select * from vw_client ; -- view

-- In real life the sensitive information such as SSN are masked,
-- In financial world Lot of attributes of Mutual Funds are maksed
-- Such as positions it is holding, What was bought and what was sold, what are the different ratings of
securities
-- buy/sell/hold etc
```

```
-- Policy_context and Conditional masking ---

-- In the session so far we have been setting the roles before doing a select to validate the masking policy
-- policy_context simulates the query results
use role accountadmin ;
use warehouse compute_wh ;
use database training ;
use schema demo ;

select current_role() ;

execute using policy_context (current_role=>'SUPPORT') as select * from client ;
execute using policy_context (current_role=>'ANALYST') as select * from client ;
execute using policy_context (current_role=>'SUPPORT') as select * from vw_client ;
execute using policy_context (current_role=>'ANALYST') as select * from vw_client ;
--

--conditional masking policy

show masking policies ;
-- lets drop the
drop masking policy pcy_text;  -- gives error and can not be dropped

-- Hence We need to unset it first, we check the objects on which the policy is currently apploed
select * from table(information_schema.policy_references(policy_name=>'pcy_text')) ;
-- unset the policy
alter table client modify column last_name unset masking policy ;
alter table client modify column ssn unset masking policy ;
alter table client modify column ethinicity unset masking policy ;
alter table client modify column status unset masking policy ;

alter view  vw_client modify column first_name unset masking policy ;
alter view  vw_client modify column street_address unset masking policy ;
```

```
-- check again
select * from table(information_schema.policy_references(policy_name=>'pcy_text')) ; -- no results
--show policies ;
show masking policies ;

-- drop -
drop masking policy pcy_text;


-- create conditional masking policy
select * from client ;
create or replace masking policy pcy_ssn as (ssn varchar, status varchar) returns varchar ->
case when current_role() in ('ANALYST') and status = 'ACTIVE' then '##SSN#MASKED#' else ssn end ;
-- first column passed should be the one to be masked
-- second column is conditional
-- apply the policy -- pay attention to the using clause while applying the policy
alter table client modify column ssn set masking policy pcy_ssn using (ssn,status);
alter view  vw_client modify column ssn set masking policy pcy_ssn using (ssn,status);
-- Validate the policy
execute using policy_context (current_role=>'SUPPORT') as select * from client ;
execute using policy_context (current_role=>'ANALYST') as select * from client ;
execute using policy_context (current_role=>'SUPPORT') as select * from vw_client ;
execute using policy_context (current_role=>'ANALYST') as select * from vw_client ;


-- cleanup
drop table client ;
drop view vw_client ;
drop masking policy pcy_ssn ;
show masking policies ;
```

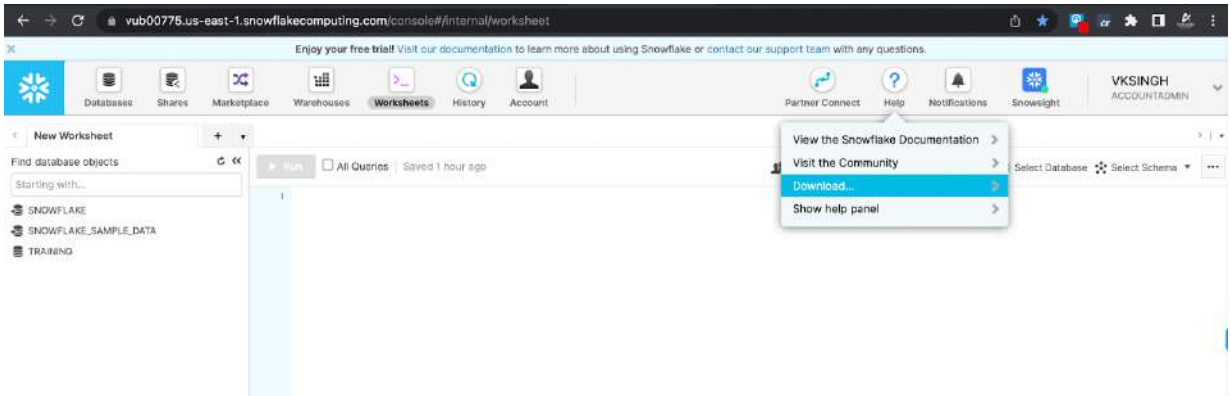
DATA VISUALIZATION

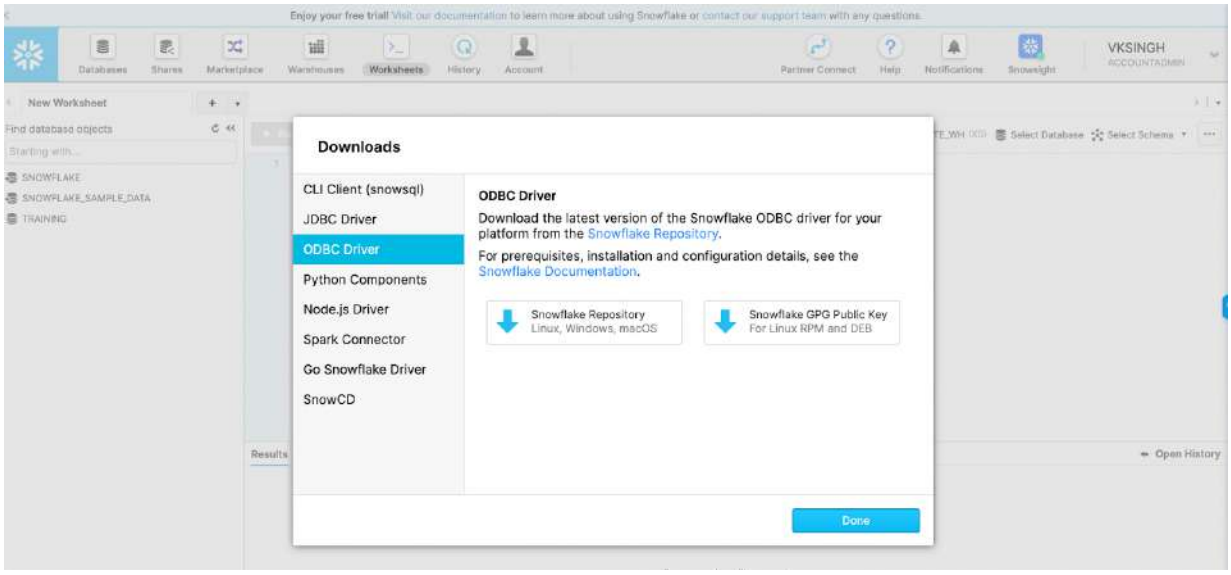
```
--- DATA VISUALIZATION -- SNOWSIGHT -- DASHBOARD VISUALIZATION --
create or replace table stores as
select s_store_sk as store_sk, s_store_id, s_store_name, s_number_employees, s_manager, s_state,
s_country
from snowflake_sample_data.tpcds_sf100tcl.store ;

select * from stores ;

select distinct i_class , i_category, i_size, i_color
from snowflake_sample_data.tpcds_sf100tcl.item ;

select * -- i_item_sk, i_current_price, i_wholesale_cost, i_category, i_class, i_size, i_color
from snowflake_sample_data.tpcds_sf100tcl.item
where  i_class = 'womens'  and i_category = 'Shoes' and i_size = 'medium' and i_color = 'floral';


create or replace table items_women_shoes_medium as
select i_item_sk as item_sk, i_rec_start_date, i_rec_end_date, i_current_price, i_wholesale_cost,
i_category, i_class, i_size, i_color
from snowflake_sample_data.tpcds_sf100tcl.item
where  i_class = 'womens'  and i_category = 'Shoes' and i_size = 'medium' and i_rec_end_date is null;


select * from items_women_shoes_medium ;

-- These  tables take some time to get created. Hence created earlier
create or replace table sales_women_shoes_medium as
select ss.ss_item_sk as item_sk, ss.ss_store_sk as store_sk, sum(ss_quantity) as sales_quantity,
sum(ss_net_profit) as profit
from snowflake_sample_data.tpcds_sf100tcl.store_sales ss
where ss_item_sk in (167259,167533,203179,242779,251337) group by ss.ss_item_sk,
ss.ss_store_sk;

-- Setting up data to visualize
select * from stores ;
-- line x- store_sk, y- no of employess (avg)
-- bar  y- store name, add column - no of employess (sum), orientation and label x (No of employees)
and y (store names)
```

```
-- cross validate the chart with below query

select s_store_name, sum(s_number_employees) from stores group by s_store_name ;

---
select * from items_women_shoes_medium ; -- Scatter , X-Axis - Item / Price (Avg)
-- scatter can show the price range --> items_sk, ^- Price

-- Heatgrid -- columns -> size, rows -> color, cell value -> average current price,
-- label rows color, check color cells based on values

-- scorecard
-- want to see average price of medium shoes for women ~ 9.1
-- check average cost 5.55
-- cost compared to price - cost 39% below of price
-- price compared to cost ~ 65% markup in price


select item_sk, sum(sales_quantity), sum(profit) from sales_women_shoes_medium  group by item_sk;
-- Show bar here with two values
-- bar graph, x - item sk, Y - Sales and Profit (sales and profit)
-- label X - Axis : Store SK, Y-Axis : Sales and Profit

-- clean up
 drop table stores ;
 drop table items_women_shoes_medium ;
 drop table sales_women_shoes_medium ;
```

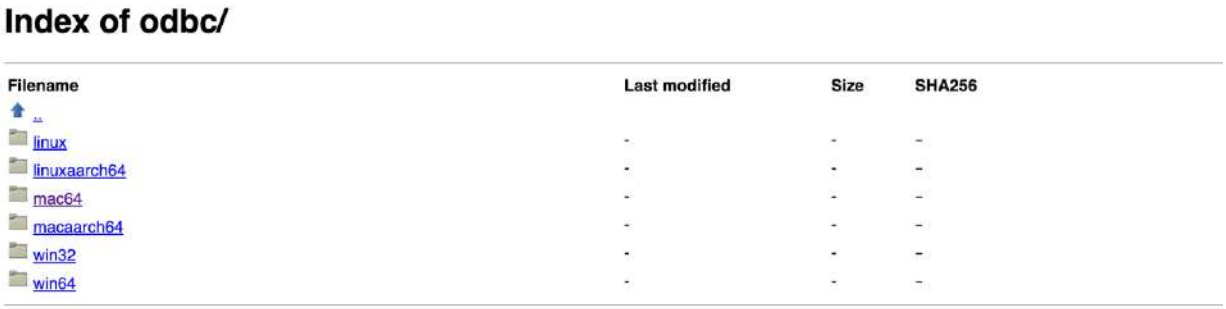
-- Using TABLEAU

– Go to snowflake Classic console
– Go to help





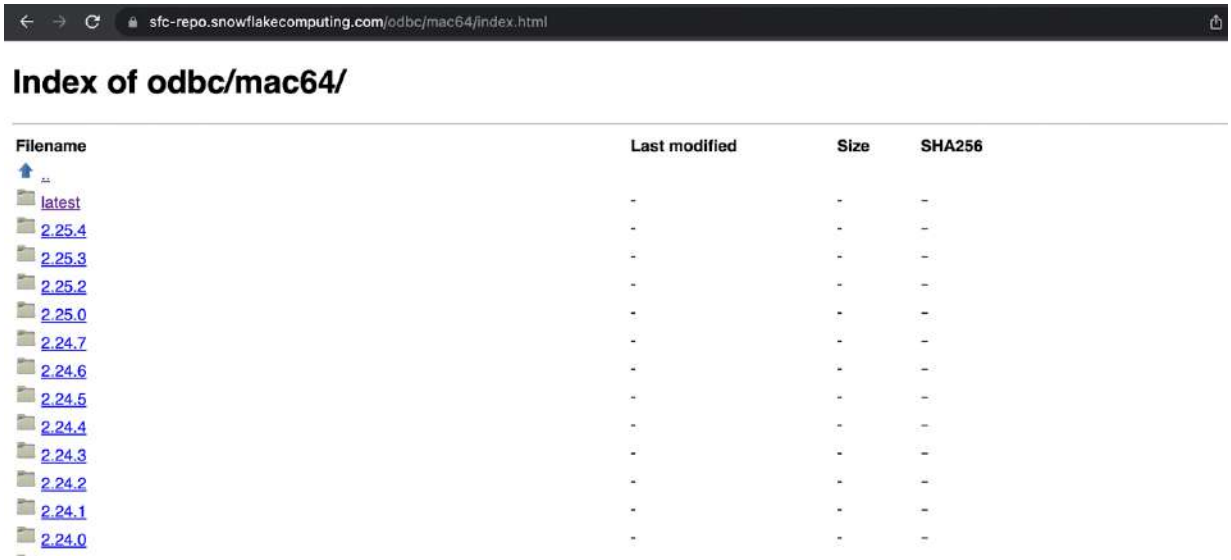
Click on Snowflake Repository



Index of odbc/

Filename	Last modified	Size	SHA256
..			
linux	-	-	-
linuxaarch64	-	-	-
mac64	-	-	-
macaarch64	-	-	-
win32	-	-	-
win64	-	-	-

Select on mac64



Click on latest

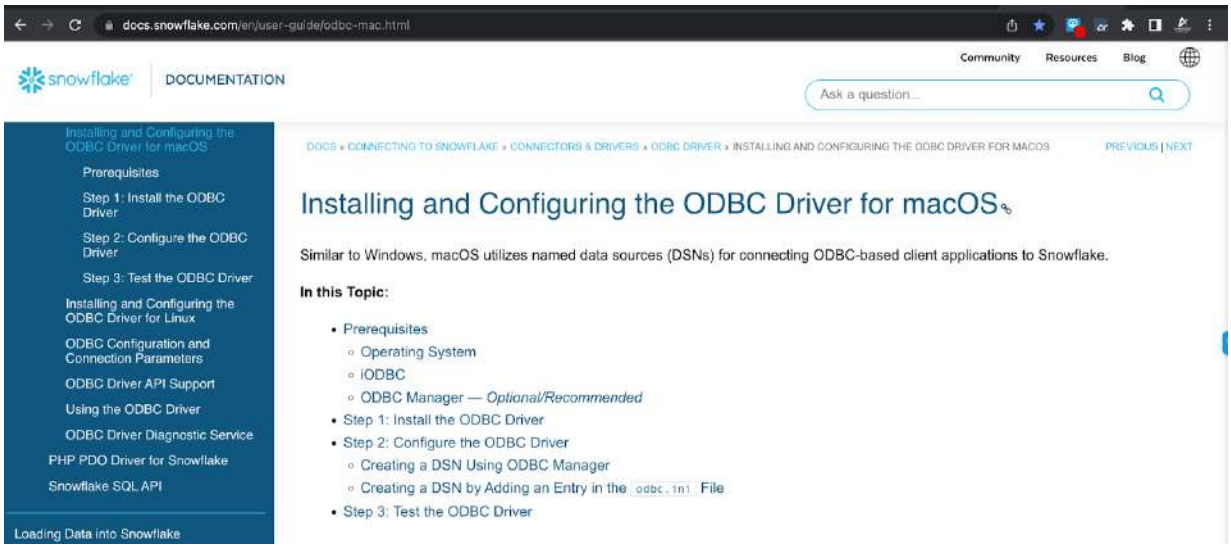
Download this and install it by clicking on it.

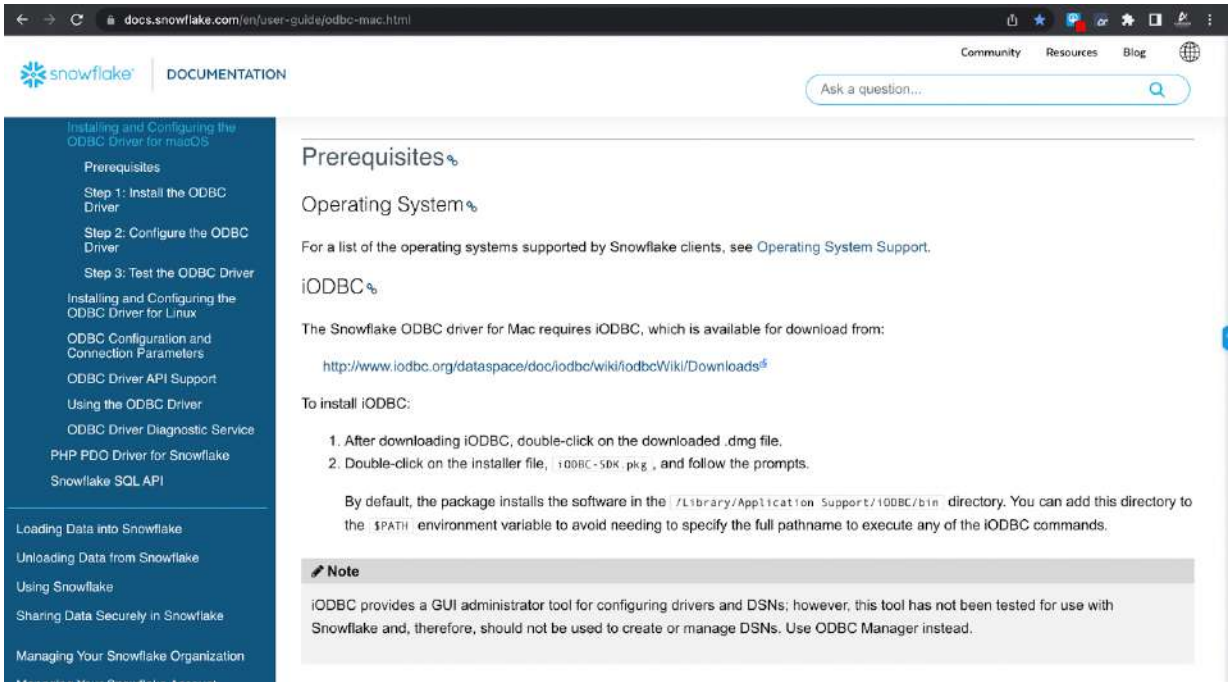
Download Tableau

https://www.tableau.com/products/desktop/download

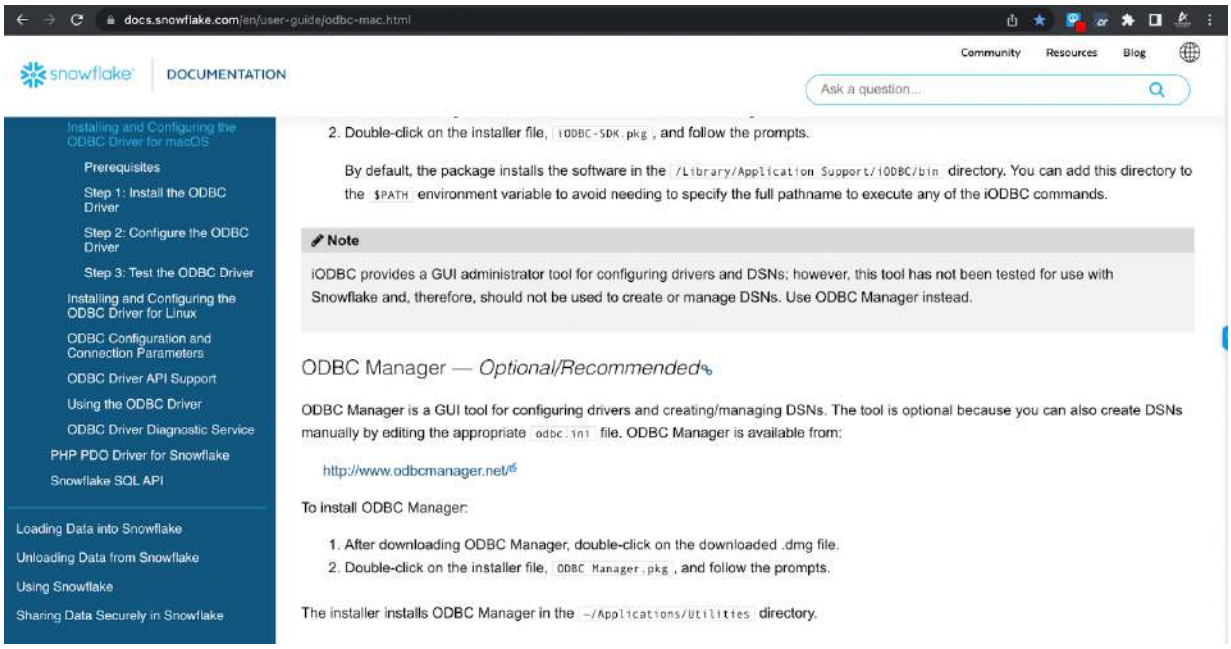
And install it.

Configure the Snowflake ODBC Driver
https://docs.snowflake.com/en/user-guide/odbc-mac.html



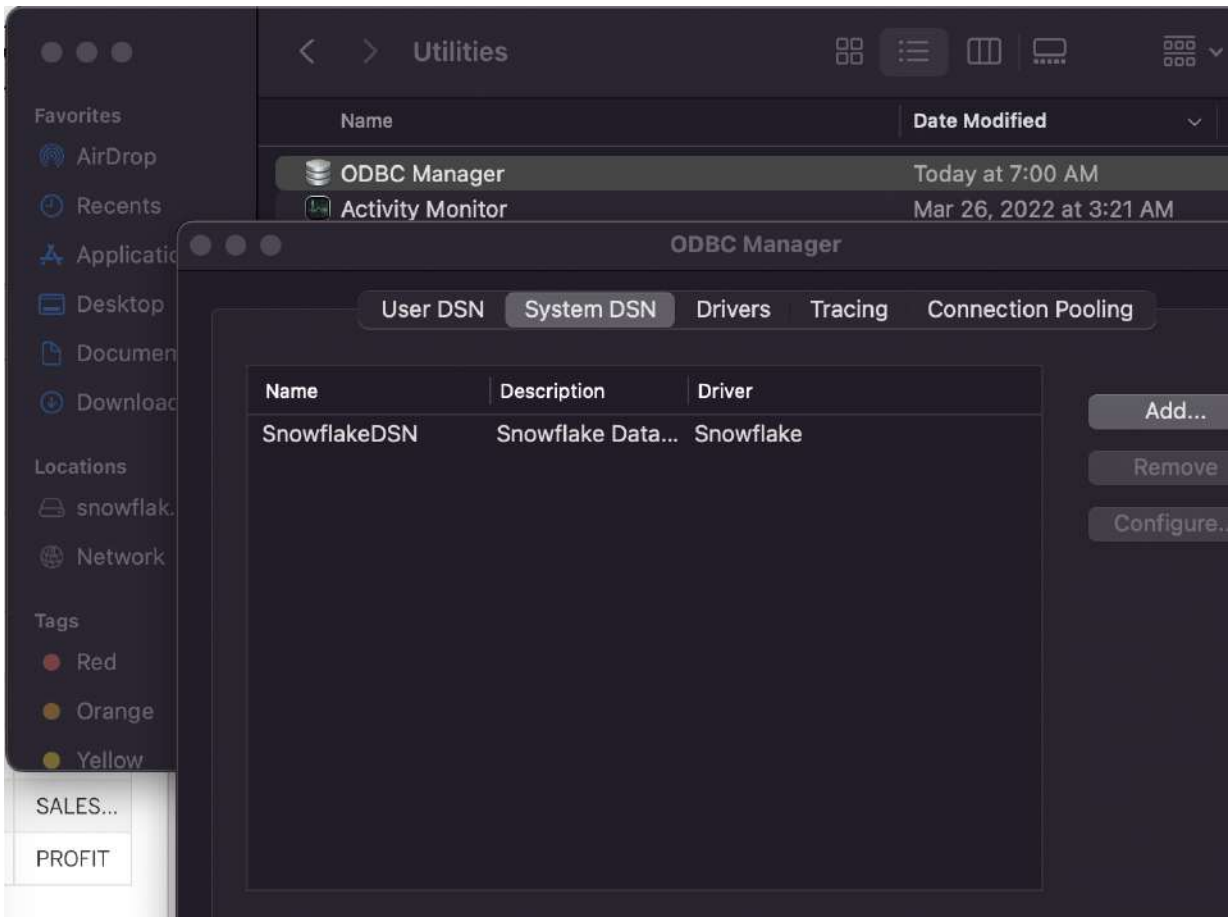


Download IODBC and install it.



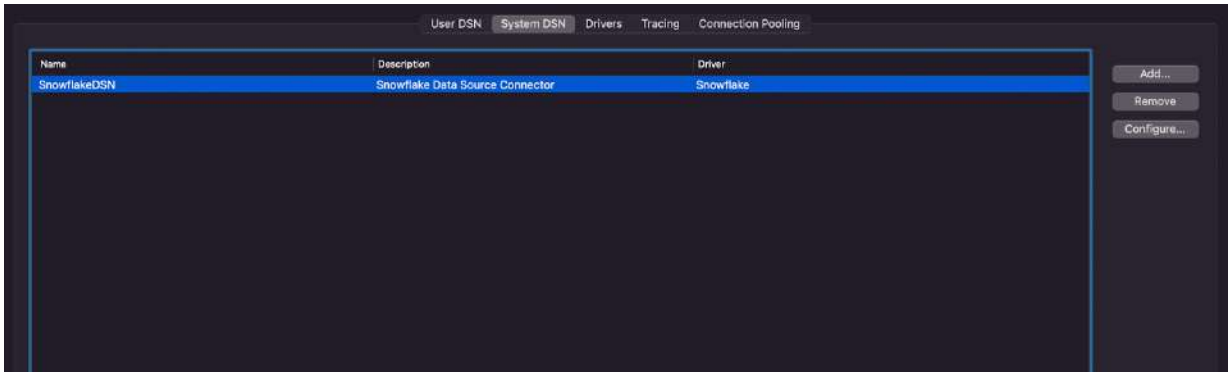
Once you download ODBC manager it will be shown in Application/Utilities

Open the ODBC manager

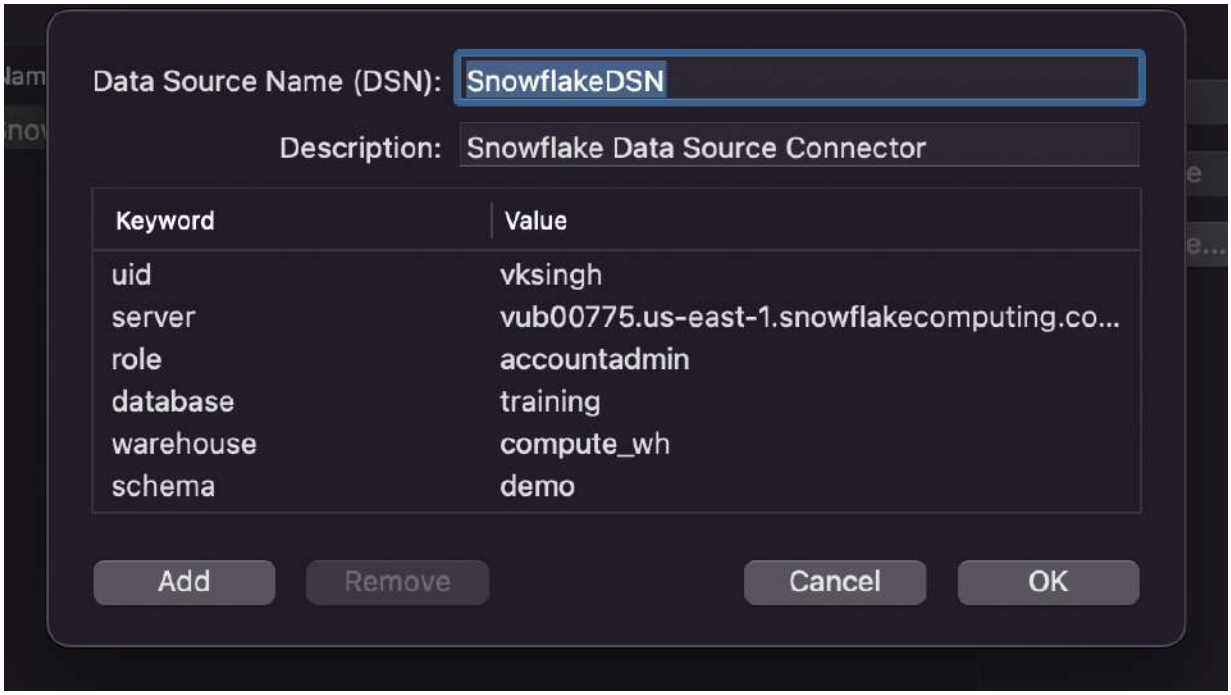


Add System DSN

Give some name and description



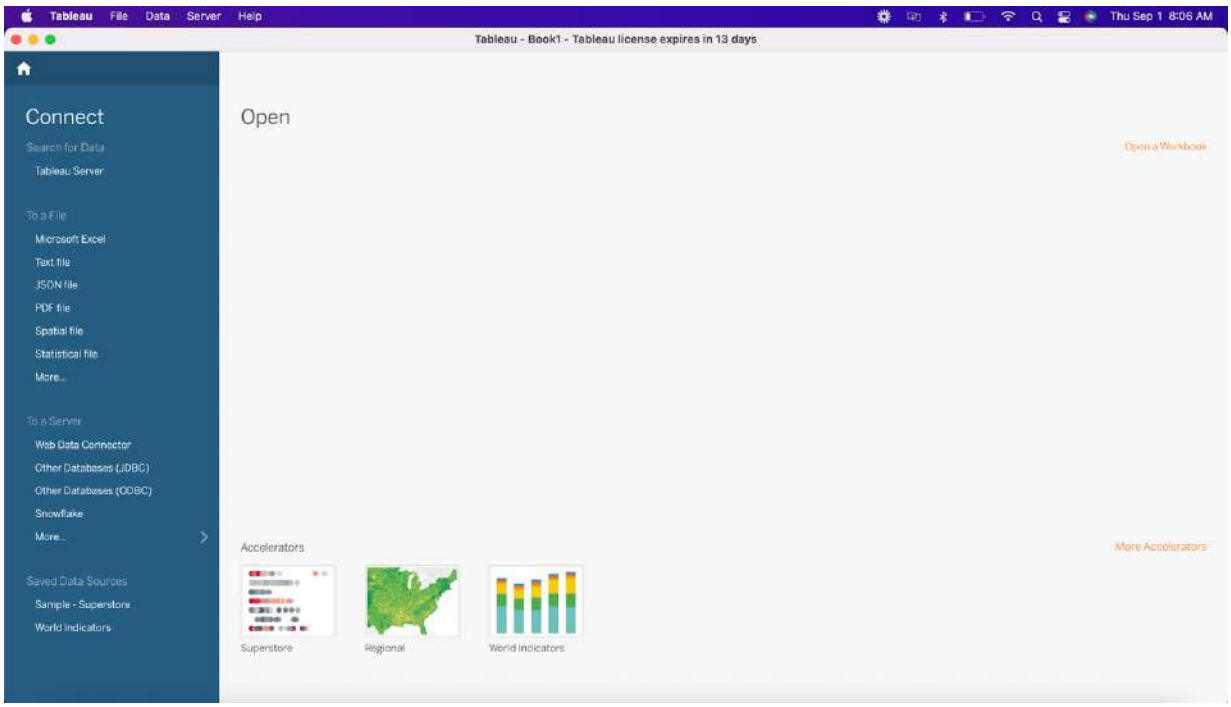
Once added - Click on configuration



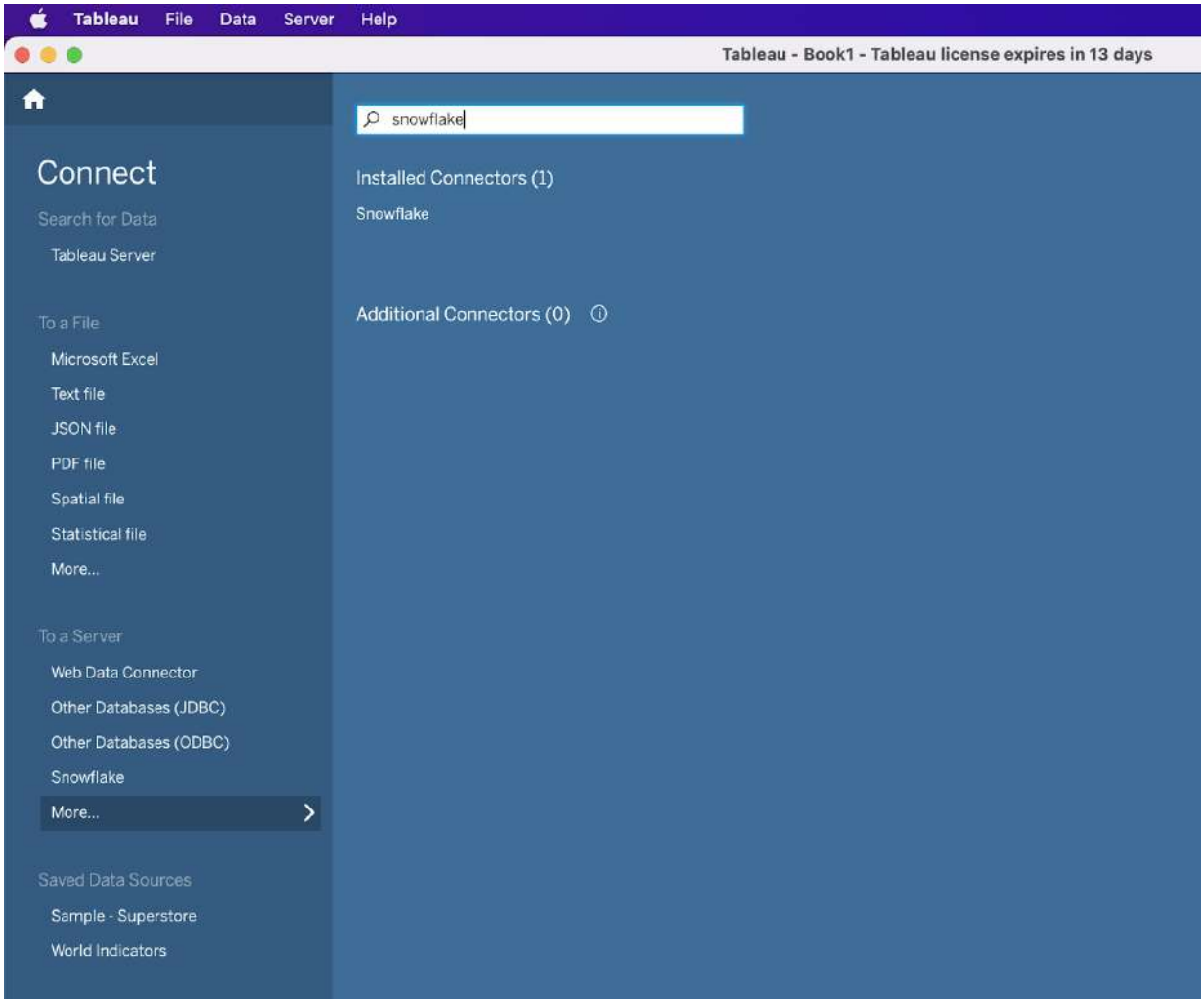
Enter Keywords and Values

Click Ok..

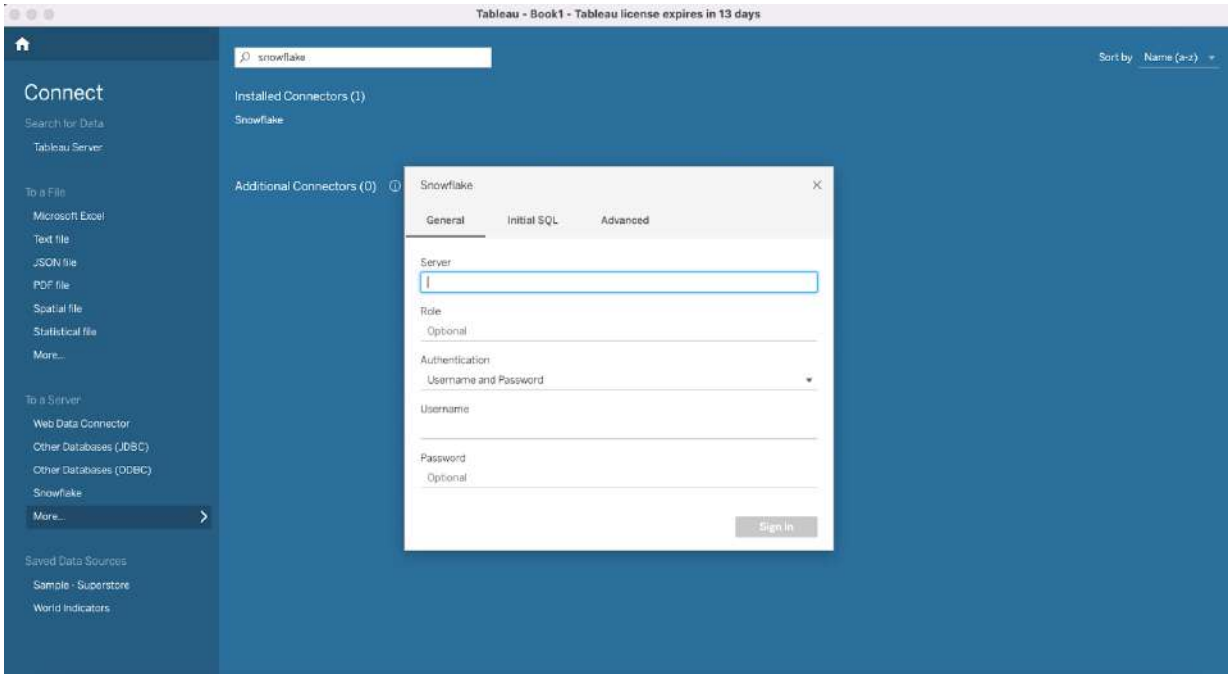
Now Go to Tableau

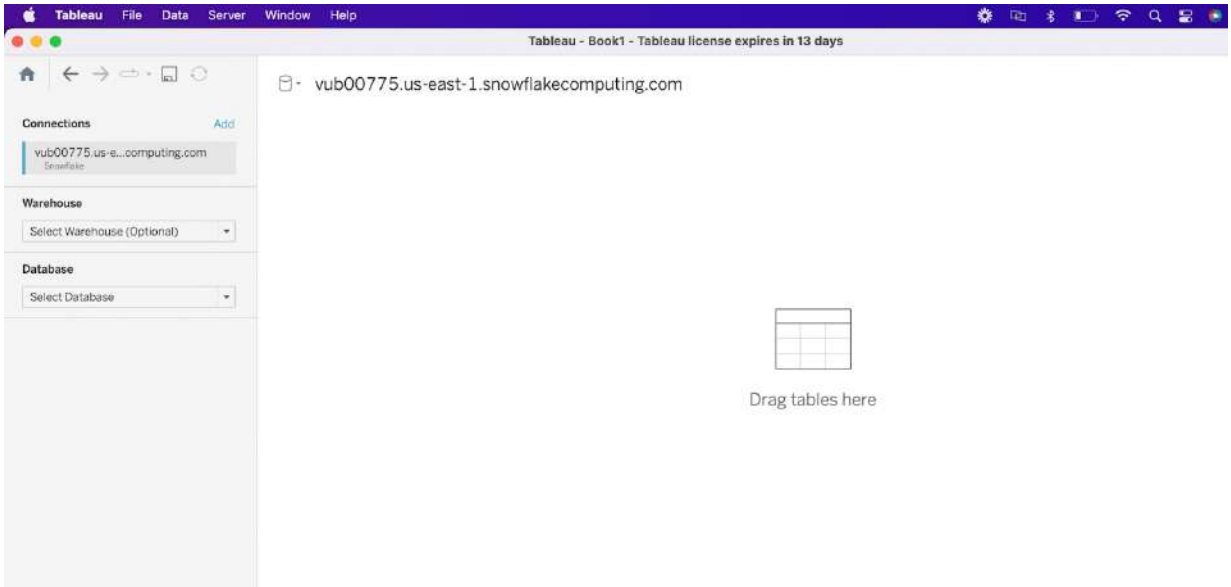


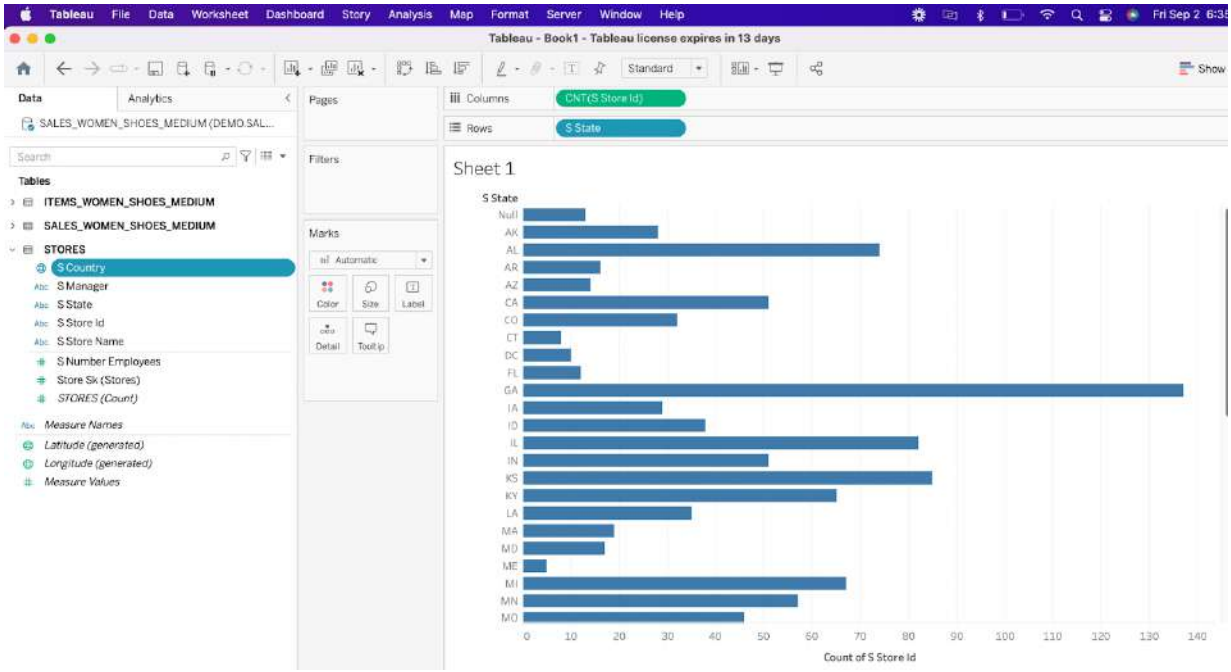
Click on Server

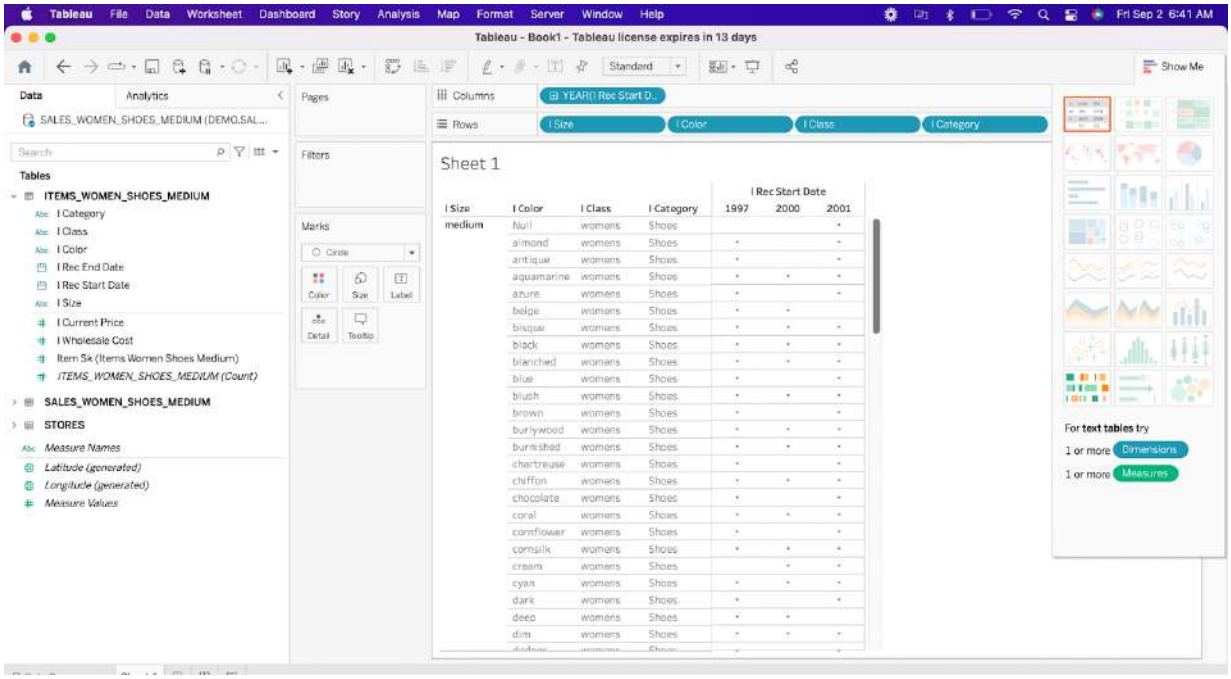


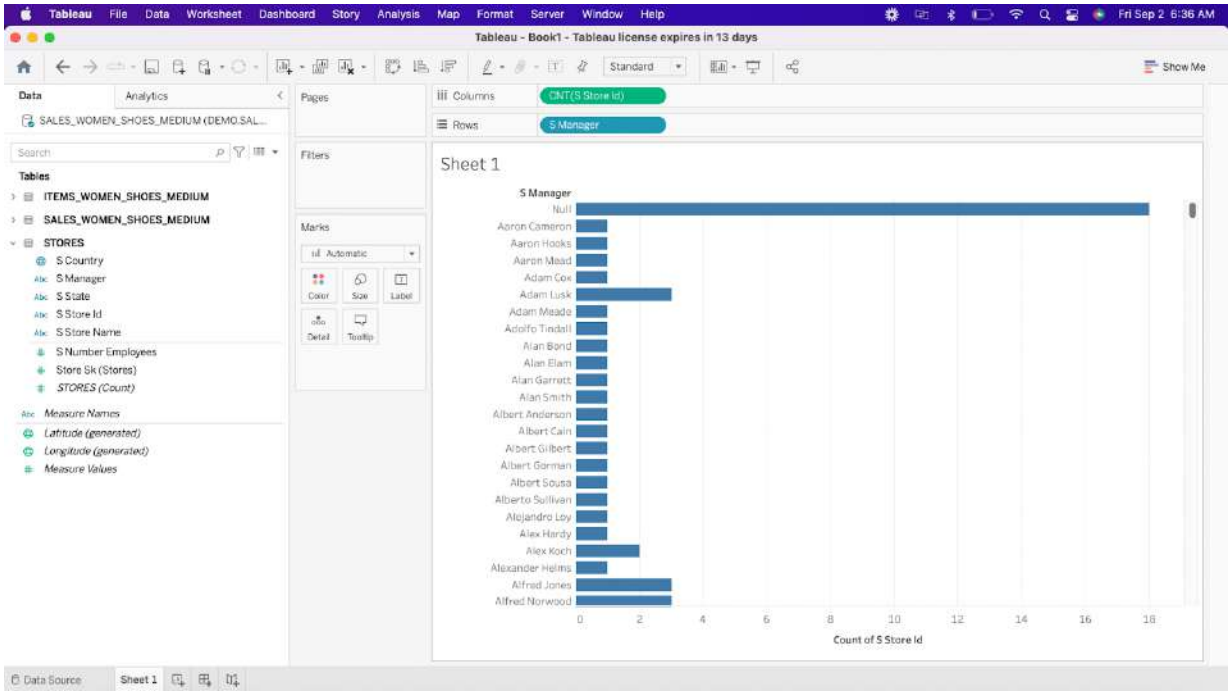
Type Snowflake

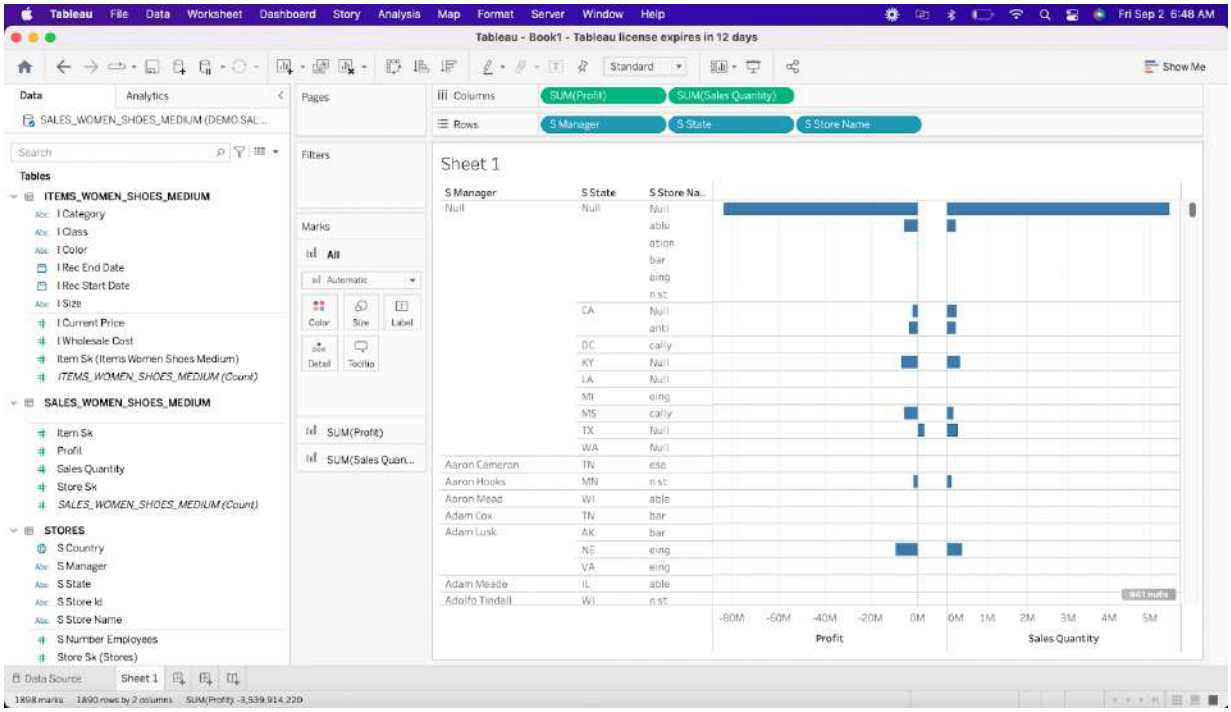


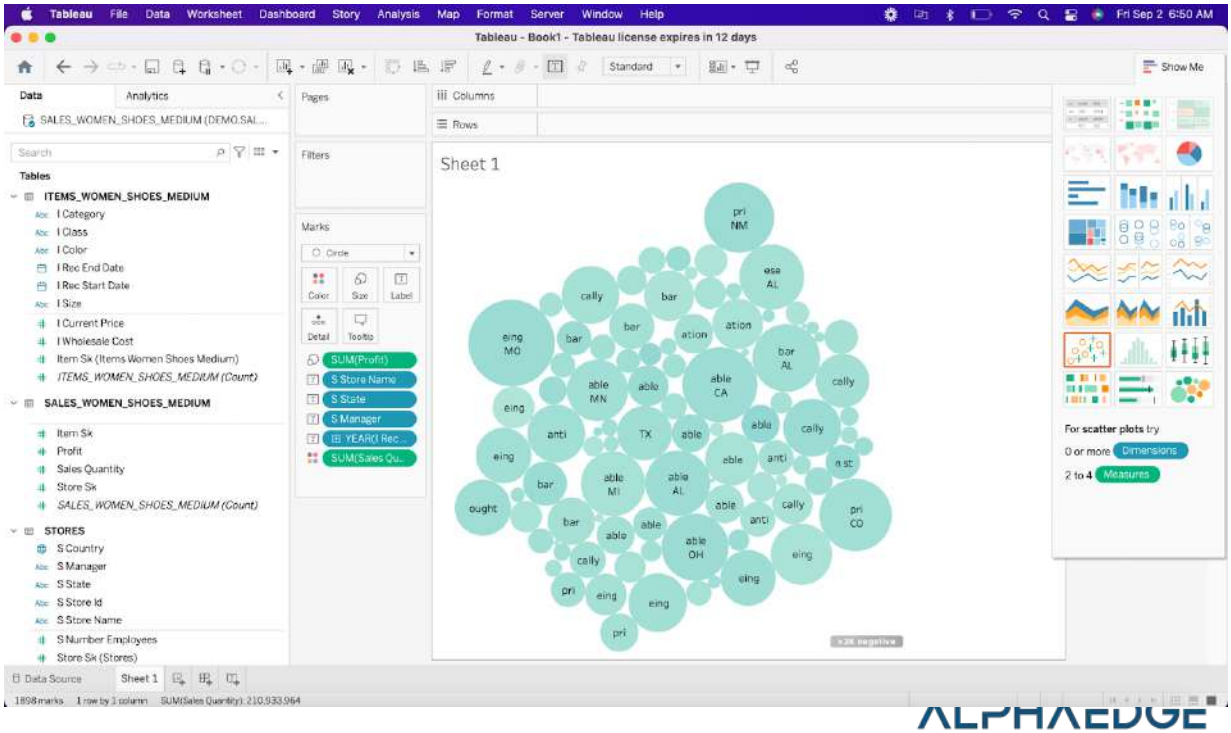












PARTNER CONNECT

1. Snowflake has agreed with some third parties or partners to connect easily and integrate.
2. This helps various organizations to integrate the tools they need for their business and connect with data in snowflake
3. Terms and conditions dictated by partners and not by snowflake
4. There is a list of supported partners
5. ACCOUNTADMIN role is required to create a partner connect trial account with email verified at snowflake.

6. **Demo to show** , how can we connect it
    a. Once you connect - a list of objects gets created in your snowflake account
        i. PC_<partner>_DB - Database
        ii. PC_<partner>_WH - Warehouse, X-Small
        iii. PC_<partner>_ROLE
        iv. PC_<partner>_USER

7. You can choose to use the existing snowflake objects by updating the partner application preference.

8. Alter user can be used to change the password for PC_<PARTNER>_USER

9. Access should be given to PC_<partner>_ROLE, if objects are owned by role other than PUBLIC

10. TROUBLESHOOTING
    a. Connection already exists - In this case either you have created it earlier OR it was initiated by a Partner.
    b. DROP objects
    c. Contact snowflake support to clear the connection and remove the checkmark
11. Validate objects after the creation

DATA SHARING AND DATA PROTECTION

Data Sharing with Security in snowflake

Following objects can be shared -

1. Tables
2. External Tables
3. Secure Views
4. Secure Materialized Views
5. Secure UDF's

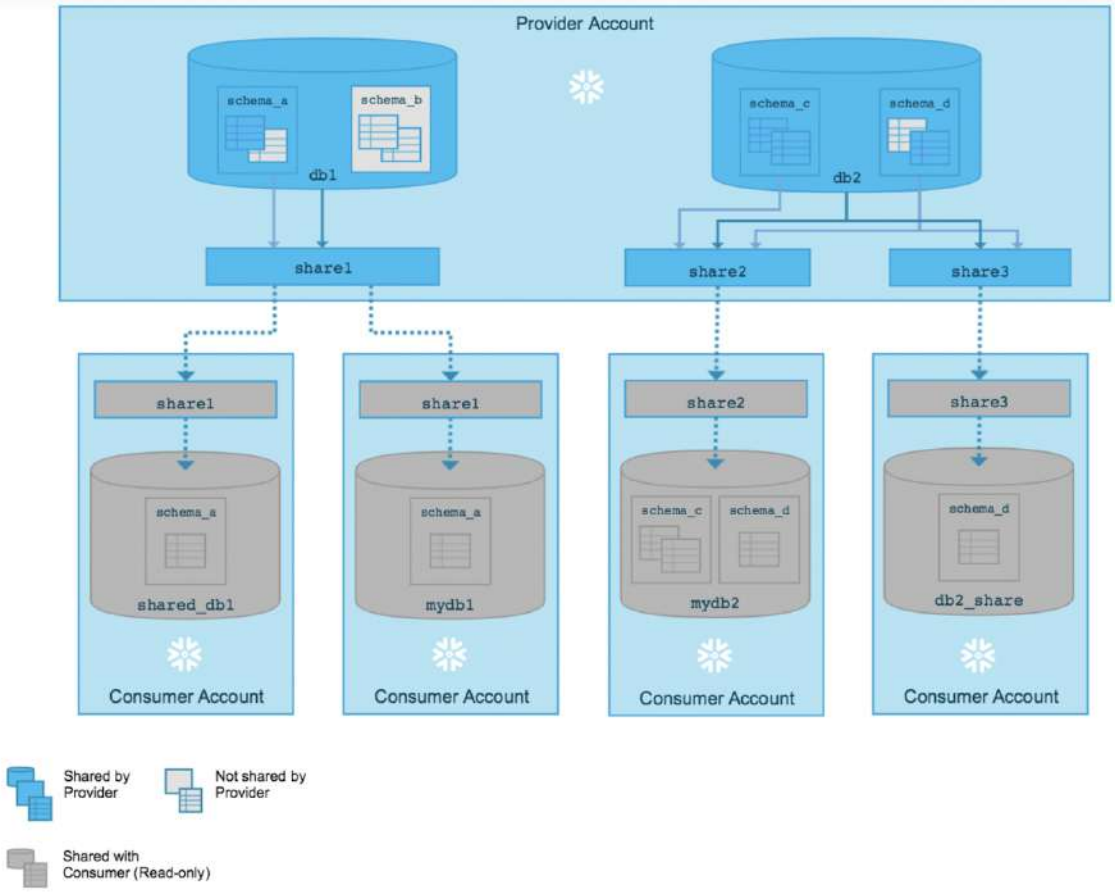
READ ONLY

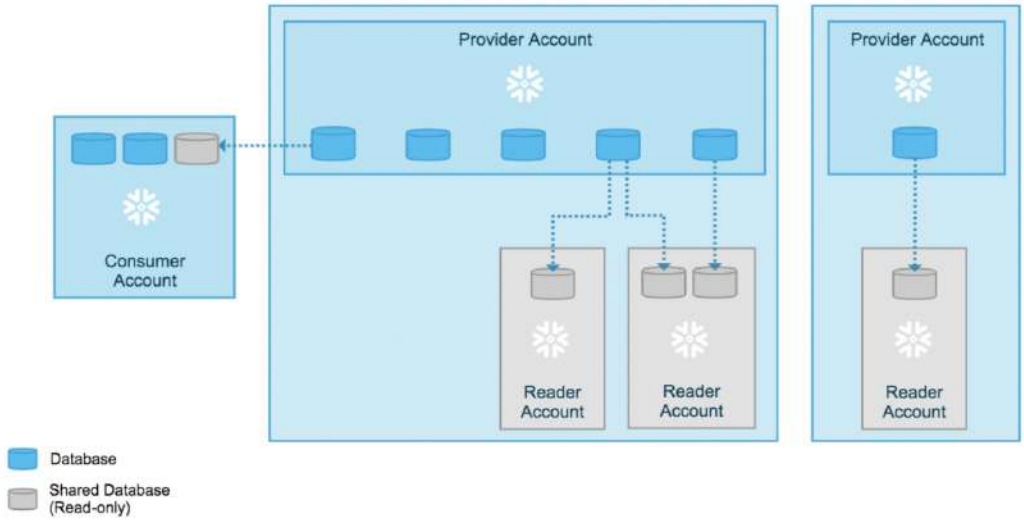
* No real data is transferred
* It is easy to setup

Snowflake achives the data sharing via "Shares". The data provider creates and give access on it to data consumers.

Share:

1. Share is DB object created in provider account
2. Grants on objects in provider account are given to share
3. Share is then modified and made available to accounts i.e consumers or readers
4. Consumer then creates database from share and access the data
5. Consumer can be another snowflake account (consumer account) or non snowflake account (Reader Account).





Data Share Demo:

PROVIDER

```
-- Give grants in warehouse and traing DB and Schema to Public
use database training ;
use schema demo ;

grant usage on warehouse compute_wh to public ;
grant usage on database training to public ;
grant usage on schema demo to public ;

create or replace storage integration int_gcp
  type = external_stage
  storage_provider = gcs
  enabled = true
  storage_allowed_locations = ('gcs://snowflakedatafiles789/csvfiles/dataload')
  comment = 'This is the integration object for loading / unloading the files from GCP to Snowflake' ;

-- Describe Intergation Object
desc integration int_gcp ;
-- goyexadihx@va3-22da.iam.gserviceaccount.com


-- create file format - since it will be used in external stage and copy command, so creating is better
create or replace file format file_format_csv skip_header = 1 compression = none;
```

```
--describe file formats
describe file format file_format_csv ;

-- create external stage for loading data
create or replace stage ext_stage_ld
  url = 'gcs://snowflakedatafiles789/csvfiles/dataload'
  storage_integration = int_gcp
  file_format = file_format_csv ;

-- Show stages
show stages ;

-- list
list @ext_stage_ld ;

  -- Create the table to load
create or replace TABLE PROMOTIONS
( P_PROMO_SK NUMBER(38,0), P_PROMO_ID VARCHAR(16), P_START_DATE_SK NUMBER(38,0), P_END_DATE_SK NUMBER(38,0),
   P_ITEM_SK NUMBER(38,0), P_COST NUMBER(15,2), P_RESPONSE_TARGET NUMBER(38,0), P_PROMO_NAME VARCHAR(50),
   P_CHANNEL_DMAIL VARCHAR(1), P_CHANNEL_EMAIL VARCHAR(1), P_CHANNEL_CATALOG VARCHAR(1), P_CHANNEL_TV VARCHAR(1),
    P_CHANNEL_RADIO VARCHAR(1), P_CHANNEL_PRESS VARCHAR(1), P_CHANNEL_EVENT VARCHAR(1), P_CHANNEL_DEMO VARCHAR(1),
    P_CHANNEL_DETAILS VARCHAR(100), P_PURPOSE VARCHAR(15), P_DISCOUNT_ACTIVE VARCHAR(1)
);

select * from promotions ;

  -- Copy command
copy into promotions from @ext_stage_ld ;

  -- check if the data is loaded
select * from promotions ;

-- one way to select the data
select $1, $2, $3, $4 from @ext_stage_ld;

-- external table
create or replace external table promotions_ext
  location=@ext_stage_ld
  auto_refresh = false
  file_format  = (type=csv);

-- Check the data of external table
select metadata$filename, metadata$file_row_number, pet.* from promotions_ext pet ;
```

```
-- UDF user Defined Functions
create or replace function get_channel (promo_sk number)
   returns varchar
   as
   $$
      select p_channel_details from promotions where p_promo_sk = promo_sk
   $$
   ;

select get_channel(23) ;


-- create data share
create or replace share training_share ;

-- give usage grants
grant usage on database training to share training_share ;
grant usage on schema demo to share training_share ;
-- give select grants on table
grant select on table promotions to share training_share ;
--give grants external table
grant select on table promotions_ext to share training_share ;
-- UDF
grant usage on function get_channel(number) to share  training_share ; -- gives error
grant usage on function get_channel(number) to role  public ;

use role public ;
use database training ;
use schema demo ;

describe function get_channel (number);

use role accountadmin ;
use warehouse compute_wh ;
-- make it secure
alter function get_channel (number) set secure ;
grant usage on function get_channel(number) to share  training_share ;


-- check the grants
show grants to share training_share ;

-- Modify the share to add a snowflake account
show shares ;
--
describe share training_share ;
--
alter share training_share add account = ERB65857 ; -- consumer account
```

-- show share via UI ***

drop share training_share ;

-- create via UI and show the consumer account

-- So far we have seen how can we actually create and share the datashare using the SQL commands and User interface
-- and shared the data with snowflake account

-- Now lets create reader account which is not a snowflake account
create managed account <account_name> admin_name <adminname> admin_password <> type = reader ;

--create managed account <account_name>, admin_name = <adminname>, admin_password = <admin_password> type = reader ;
create managed account account_non_snowflake admin_name= admin_non_snowflake, admin_password = 'Welcome123!', type = reader ;

--{"accountName":"ACCOUNT_NON_SNOWFLAKE","accountLocator":"FZB10863","url":"https://bguerbp-account_non_snowflake.snowflakecomputing.com","accountLocatorUrl":"https://fzb10863.us-east-1.snowflakecomputing.com"}

show managed accounts ;

alter share training_share add account = FZB10863 share_restrictions = false;

```
-- let's create one more table and access to this table should be given to different role

-- Create a table to demonstrate
create or replace table client
(  id NUMBER(38,0), first_name VARCHAR(16), last_name VARCHAR(50),
   sex VARCHAR(1), ethinicity VARCHAR(30), ssn VARCHAR(15),
   street_address VARCHAR(90),status VARCHAR(10)
);

-- insert some data
delete from client ;
insert into client values (111111, 'James',      'Schwartz', 'M', 'American','342-76-9087','5676 Washington Street','ACTIVE') ;
insert into client values (222222, 'Jessica',    'Escobar',  'F', 'Hispanic','456-93-5629','3234 WateringCan Drive','INACTIVE') ;
insert into client values (333333, 'Ben',        'Hardy',    'M', 'American','876-98-3245','6578 Historic Circle','INACTIVE') ;
insert into client values (444444, 'Anjali',     'Singh',    'F', 'Indian American','435-87-6532','8978 Autumn Day Drive','ACTIVE') ;
insert into client values (555555, 'Dean',       'Tracy',    'M', 'African','767-34-7656','2343 India Street','ACTIVE') ;

select * from client ;

---
grant select on table client to share training_share ;



-------------------------------------------------------

-- share entire database and schema
drop share training_share ;
-- create a share for entire DB
create share training_db_share ;


-- give usage grants
grant usage on database training to share training_db_share ;
grant usage on schema demo to share training_db_share ;

-- Lets create one more table
create or replace table items as select * from SNOWFLAKE_SAMPLE_DATA.TPCDS_SF100TCL.ITEM ;
-- check tables
show tables ;

-- give grants
```

```
grant select on all tables in database training to share training_db_share ;
grant select on all external tables in database training to share training_db_share ;
grant usage on all functions in database training to share training_db_share ; -- error

grant usage on function get_channel(number) to share training_db_share ; -- Hence need to give specific

-- Add snowflake consumer account and non snowflake reader account
alter share training_db_share add account = ERB65857 ; -- snowflake consumer account, non reader account

alter share training_db_share add account = FZB10863  share_restrictions = false;

show shares ;

-- Go to consumer and reader account



-----------------------------------------------

-- client -- operations
select * from client ;

-- check insert / update / delete operations effect
insert into client values (666666, 'Shawn', 'Pollock', 'M', 'American','457-34-4532','2343 Washington Street','ACTIVE') ;
delete from client where id = 666666 ;
update client set status = 'INACTIVE' where id = 555555 ;

-- new table creation
create or replace table income_band as select * from SNOWFLAKE_SAMPLE_DATA.TPCDS_SF100TCL.INCOME_BAND ;
--data
select * from income_band ;
-- grant again
grant select on income_band to share training_db_share ;

-- check in consumer and reader account

-- new view creation
create or replace view vw_client as select * from client where sex = 'M' ;
select * from vw_client ;

-- grant is needed
grant select on vw_client to share training_db_share ;
grant select on vw_client to role public ;
```

```
-- show view
show views ;


-- secure view
create or replace secure view vw_client_sec as select * from client where sex = 'M' ;
grant select on vw_client_sec to role public ;
select * from vw_client_sec ;

-- show view
show views ;

-- grant is needed
grant select on vw_client_sec to share training_db_share ;


-- create a materialized secure view and show
create materialized view mvw_client
   comment='Materialized View on table client'
   as
   select * from client where sex = 'F' ;

select * from mvw_client ;

--describe
describe materialized view mvw_client ;
-- show
show views ;

grant select on mvw_client to role public ;

-- set the role public and check
use role public ;
use warehouse compute_wh ;
use database training ;
use schema demo ;
--show views
show views ;
--set accountadmin
use role accountadmin ;
use warehouse compute_wh ;
use database training ;
use schema demo ;
-- set secure
alter materialized view mvw_client set secure ;
-- unset secure
alter materialized view mvw_client unset secure ;
-- grant materialized view to share
grant select on mvw_client to share training_db_share ;
```

```
-- check in customer


-- Simulated data sharing
alter session set simulated_data_sharing_consumer=wRB65857;
select * from mvw_client;





  -- Cleanup
drop file format if exists file_format_csv ;
drop stage if exists ext_stage_ld ;
drop table if exists promotions ;
drop share if exists training_share ;
drop table if exists client ;
drop share if exists training_db_share ;
drop view if exists vw_client ;
drop view if exists vw_client_sec ;
drop table if exists income_band ;
drop table if exists items ;
drop function if exists get_channel(number) ;
drop managed account if exists account_non_snowflake ;
```

CONSUMER

```
-- Go to Snowflake consumer Account
show shares ;
-- <organization>.<account>.<share>

-- The above command works at an account level.

-- Now we need to create a database using the share
create database training from share BGUERBP.AYB85365.TRAINING_SHARE ;

-- use database
use database training ;
use schema demo ;
```

```
-- check the data in the table share via share
select * from promotions ;
-- external table
select metadata$filename, metadata$file_row_number, pet.* from promotions_ext pet ;
-- UDF
select get_channel(23) ;

describe function get_channel (number);
```

```
select * from client ;



-- sharing the entire database queries
drop database training ;
show databases;
-- use other DB
use database snowflake ;
show shares ;

-- drop
drop database training_db ;
-- create database
create database training_db from share BGUERBP.AYB85365.TRAINING_DB_SHARE ;

-- use database
use database training_db ;
use schema demo ;

-- client
select * from client ;
-- income_band
select * from items ;
-- promotions
select * from promotions ;
-- promotions_ext
select * from promotions_ext ;
--get_channel(24)
select get_channel(24) ;
```

```
-- show views ;
select * from client ;
select * from income_band ;
select * from vw_client_sec ;


-- views
select * from vw_client_sec ;
select * from mvw_client;


update client set status = 'INACTIVE' where id = 111111 ;



-- cleanup
drop database training ;
drop database training_db ;
```

READER

```
-- non snowflake reader account

use role accountadmin ;


show shares ;
-- <organization>.<account>.<share>

-- The above command works at an account level.

-- Now we need to create a database using the share
create database training from share BGUERBP.AYB85365.TRAINING_SHARE ;

-- use database
use database training ;
use schema demo ;
--
-- Need to create the compute power via warehouse
```

```
create warehouse non_snowflake_wh with warehouse_size = 'x-small' auto_suspend = 100
auto_resume = true ;
--
use warehouse non_snowflake_wh ;
--
-- check the data in the table share via share
select * from promotions ;
-- external table
select metadata$filename, metadata$file_row_number, pet.* from promotions_ext pet ;
-- UDF
select get_channel(23) ;

describe function get_channel (number);




---------------------------------------------
-- To show imported privilege now we have access to two tables
select * from client ;

-- create roles
create or replace role support ;
create or replace role analyst ;

grant role analyst to user admin_non_snowflake ;
grant role support to user admin_non_snowflake ;

grant usage on warehouse non_snowflake_wh to role analyst ;
grant usage on warehouse non_snowflake_wh to role support ;

-- "IMPORTED" privileges

-- grant schema usage to roles
grant usage on database training to role support;  -- error
grant usage on database training to role analyst;  -- error

grant imported privileges on database training to role support;
grant imported privileges on database training to role analyst;

--
use role analyst ;
use warehouse non_snowflake_wh ;
select * from promotions ;
-- external table
select metadata$filename, metadata$file_row_number, pet.* from promotions_ext pet ;
-- UDF
select get_channel(23) ;
--client
```

```
select * from client ;


--
use role accountadmin ;
revoke imported privileges on database training from role analyst;

use role analyst ;
use warehouse non_snowflake_wh ;
select * from promotions ;
-- external table
select metadata$filename, metadata$file_row_number, pet.* from promotions_ext pet ;
-- UDF
select get_channel(23) ;
--client
select * from client ;









-- sharing the entire database and schema
use role accountadmin ;
use database snowflake ;
-- show shares
show shares ;

-- Now we need to create a database using the share
create database training_db from share BGUERBP.AYB85365.TRAINING_DB_SHARE ;

-- use database
use warehouse non_snowflake_wh ;
use database training_db ;
use schema demo ;

select * from promotions ;
-- external table
select metadata$filename, metadata$file_row_number, pet.* from promotions_ext pet ;
-- UDF
select get_channel(23) ;
--client
select * from client ;
--items
select * from items ;
```

```
--------

-- check data
select * from client ;
select * from income_band ;
select * from vw_client_sec ;

-- show views ;
show views ;

-- views
select * from vw_client_sec ;
select * from mvw_client;

-- update operations are not supported in reader account
update client set status = 'INACTIVE' where id = 111111 ;

-- Cleanup
use role accountadmin ;
drop role analyst ;
drop role support ;
drop database training ;
drop warehouse non_snowflake_wh ;
drop database training_db ;
```

TIME TRAVEL

- It is the feature which allows a user to access the data back in history with a defined period - **RETENTION PERIOD ~ 1 - 90 days, Default 1**
- Time travel helps to protect data from loss of data due to (intentional or unintentional) update or delete operations.
- Protects from data errors resulting during various operations (writing, processing, batch execution,production installs) due to which unwanted changes can also go in the database.

- Features:
    - Can go in the past and retrieve/select data via
        - Exact timestamp in the past
        - Using offset e,g in last few hours
        - Using query ID. example - get the data before a query execution.
        - Select data using time travel **AT | BEFORE**
        - Clone data using time travel **AT | BEFORE**

    - Restores tables, schemas and databases.
        - UNDROP

How time travel works :

Diagram is taken from Snowflake documentation
https://docs.snowflake.com/en/user-guide/data-time-travel.html

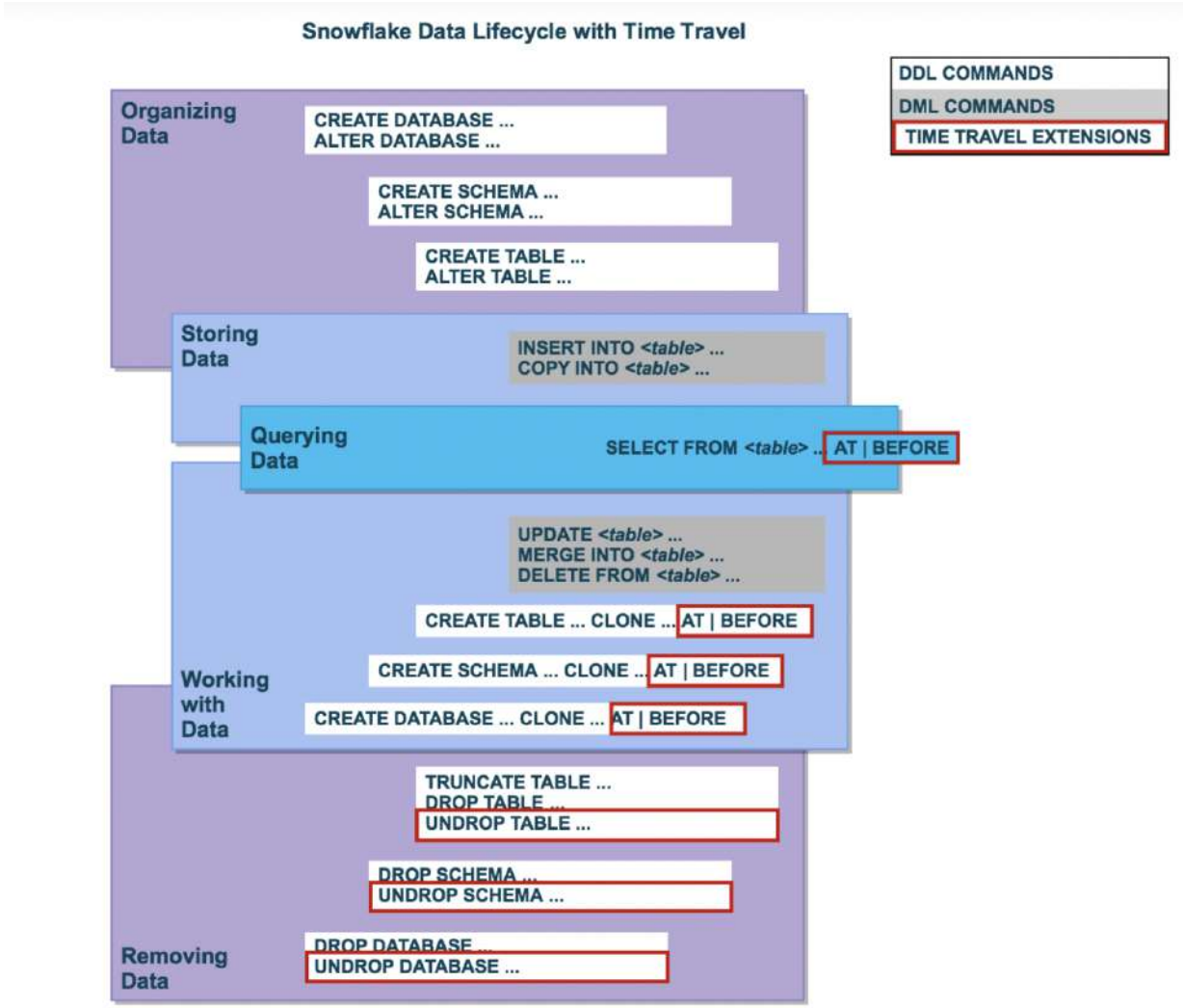
Demo:

```
show parameters like '%retention%' in account; -- default is 1 and 0, Value can be 1 - 90 days

alter account set DATA_RETENTION_TIME_IN_DAYS = 1 ;
alter account set MIN_DATA_RETENTION_TIME_IN_DAYS = 2 ;

-- effective retention time = max (data_retention_time_in_days, min_data_retention_time_in_days)

alter database set DATA_RETENTION_TIME_IN_DAYS = 5 ; --3
alter database set MIN_DATA_RETENTION_TIME_IN_DAYS = 1 ; -- error

show parameters like '%retention%' in database; -- only 1



alter schema set DATA_RETENTION_TIME_IN_DAYS = 6 ;
alter schema set MIN_DATA_RETENTION_TIME_IN_DAYS = 1 ; --error

show parameters like '%retention%' in schema; -- only 1 ;
```

```
-- create tables for time travel
create or replace table income_band as select * from
SNOWFLAKE_SAMPLE_DATA.TPCDS_SF100TCL.INCOME_BAND ;

-- Create table client
create or replace table client
(  id NUMBER(38,0), first_name VARCHAR(16), last_name VARCHAR(50),
   sex VARCHAR(1), ethinicity VARCHAR(30), ssn VARCHAR(15),
   street_address VARCHAR(90),status VARCHAR(10)
);


delete from client ;
insert into client values (111111, 'James',     'Schwartz', 'M', 'American','342-76-9087','5676 Washington
Street','ACTIVE') ;
insert into client values (222222, 'Jessica',   'Escobar',  'F', 'Hispanic','456-93-5629','3234 WateringCan
Drive','INACTIVE') ;
```

```
insert into client values (333333, 'Ben',       'Hardy',    'M', 'American','876-98-3245','6578 Historic
Circle','INACTIVE') ;
insert into client values (444444, 'Anjali',     'Singh',   'F', 'Indian American','435-87-6532','8978 Autumn
Day Drive','ACTIVE') ;
insert into client values (555555, 'Dean',      'Tracy',    'M', 'African','767-34-7656','2343 India
Street','ACTIVE') ;

-- select data
select * from income_band ; -- 20 rows
select * from client ; -- 5 rows

-- Time travel selects
show parameters like '%time%' ;
alter session set timezone = 'America/New_York'; -- Snowflake does not support EST, EDT. It supports
->America/Los_Angeles, Europe/paris, Asia/Tokyo

select current_timestamp() ;
-- <>
-- Do some deletes and updates on both the tables save query ID
-- delete
delete from income_band where ib_lower_bound > 101000;  -- accidental delete, wanted to delete >
121000
-- Query ID
-- <>
-- update
update client set status = 'INACTIVE' ;                -- accidental update, shouyld have been only for
222222 ;
-- Query ID
-- <>


-- see the data changes
select * from income_band ; -- 11 rows, wanted rows for 110 and 120
select * from client ; -- All are INACTIVE


-- How can we query using time travel BEFORE | AT

-- using before via timestamp
select * from client before (timestamp => ''::timestamp_ltz); -- shows 5 rows with correct status
select * from income_band before (timestamp => ''::timestamp_ltz); -- shows all 20 rows

--using before via query ID
select * from income_band before (statement => '');
select * from client before (statement => '');

-- get the seconds to be used for offset
select datediff(second, ''::timestamp_ltz, current_timestamp()) as seconds;
```

```
-- The offset is in seconds using at
select * from client at(offset => -);

-- get right seconds
select datediff(second, ''::timestamp_ltz, current_timestamp()) as seconds;

-- show how the offset can be used
select * from income_band at(offset => -);
--
```



Note: Diagram is taken from Snowflake documentation
https://docs.snowflake.com/en/user-guide/data-time-travel.html

Data Restoration:

Direct - This drops and recreates the table

Intermediary table - This method has more flexibility.

Demo:

```
-- Data Restoration Direct Method--Has Limitations
-- Create table client
create or replace table client
(  id NUMBER(38,0), first_name VARCHAR(16), last_name VARCHAR(50),
   sex VARCHAR(1), ethinicity VARCHAR(30), ssn VARCHAR(15),
   street_address VARCHAR(90),status VARCHAR(10)
);


delete from client ;
insert into client values (111111, 'James',     'Schwartz', 'M', 'American','342-76-9087','5676 Washington
Street','ACTIVE') ;
insert into client values (222222, 'Jessica',   'Escobar',  'F', 'Hispanic','456-93-5629','3234 WateringCan
Drive','INACTIVE') ;
insert into client values (333333, 'Ben',       'Hardy',    'M', 'American','876-98-3245','6578 Historic
Circle','INACTIVE') ;
insert into client values (444444, 'Anjali',    'Singh',    'F', 'Indian American','435-87-6532','8978 Autumn
Day Drive','ACTIVE') ;
insert into client values (555555, 'Dean',      'Tracy',    'M', 'African','767-34-7656','2343 India
Street','ACTIVE') ;

-- select data
select * from client ; -- 5 rows


-- update
update client set status = 'INACTIVE' ; -- accidental update
-- queryid
-- <>
select * from client before (statement => '');
select * from client;

-- delete
delete from client  ;                -- accidental delete
-- Query ID
-- <>
```

```
select * from client before (statement => '');
select * from client;

-- Direct data restoration
create or replace table client as select * from client before (statement => '');

-- check the data
select * from client ;

-- Direct data restoration to previous Query before the update, errors and hence the limitation.
create or replace table client as select * from client before (statement => '');
```

```
----INDIRECT METHOD --No Limitation
-- Lets try the Indirect method
-- create the table
create or replace table income_band as select * from
SNOWFLAKE_SAMPLE_DATA.TPCDS_SF100TCL.INCOME_BAND ;
-- First accidental delete
delete from income_band where ib_lower_bound > 101000;  -- accidental delete, wanted to delete >
121000
-- query_id
-- <01a78be3-0004-2d64-0026-d98700065126>
select * from income_band ;
-- Second accidental update
update income_band set ib_lower_bound = ib_lower_bound + 1000, ib_upper_bound =
ib_upper_bound + 1000 ; -- shift the band by 100, accidentally shifted by 1000.
-- query_id
-- <01a78be4-0004-2d7d-0026-d9870006f05e>
-- check data
select * from income_band ;
```

```
--Using indirect method latest update - using a temporary
create or replace table income_band_temp as  select * from income_band before (statement => '');

-- check data in temp
select * from income_band_temp ;
select * from income_band ;

-- delete the main table
delete from  income_band
-- restore from temp table
insert into income_band select * from income_band_temp;
-- check data
select * from income_band ; -- we went back one step, Now we realize that we need to go back further

-- Again we can build the temp table using first query , lets see
create or replace table income_band_temp as  select * from income_band before (statement => '');

-- delete the main table
delete from  income_band
-- restore from temp table
insert into income_band select * from income_band_temp;
-- check data
select * from income_band ;

-- The idea is you should not drop the table to have its data retention.
-- The moment a table is dropped it loses time travel and is considered as a new object,
-- create or replace actually drops the object.
```

—----------

UNDROP:

Drop
History
Undrop

Demo:

```
-- UNDROP tables, schemas, database

show <object_type> history ;

-- if you see object via above command you can restore it via undrop

show tables history ;
show schemas history ;
show databases history ;


-- TABLE
show tables ;
-- check the data in the table
select * from promotions ;
-- drop table
drop table promotions ;
-- show tables
show tables ; -- not visible
-- show tables history
show tables history ; -- visible, Hence can be undropped
--undrop
undrop table promotions ;
-- show tables
show tables ;
select * from promotions ;

-- Schemas
show schemas ;
--
create or replace schema cln_demo clone demo ;

drop schema cln_demo;
show schemas ;
show schemas history ;
undrop schema cln_demo ;

-- clean up
drop schema cln_demo ;
```

```
-- Databases

show databases ;
--
create or replace database cln_training clone training ;

drop database cln_training;
show databases ;
show databases history ;
undrop database cln_training ;
-- clean up
drop database cln_training ;
```

Retention Time;

This is the main component of time travel. The time travel is achieved by using the value of this parameter

Account Level
Database Level
Schema Level
Object level

How:

Default is 1 day and automatically enabled.
0 means time travel is disabled.

DATA_RETENTION_TIME_IN_DAYS
MIN_DATA_RETENTION_TIME_IN_DAYS

effective:
max(DATA_RETENTION_TIME_IN_DAYS, MIN_DATA_RETENTION_TIME_IN_DAYS)

Standard	Enterprise	Business Critical	Virtual private
1 (Default), Can be set to 0 at account and object level Range 0 - 1	90 Range 0 - 90	90 Range 0 - 90	90 Range 0 - 90

```
---------- Data retention ------
-- Retention Time

create database training_rt ;
create schema demo_rt ;

---------- Data retention ------
show parameters like '%retention%' in account;
show parameters like '%retention%' in database;
show parameters like '%retention%' in schema;

create table test_client as select * from training.demo.client ;
show tables; -- retention time is 1 taken from account

alter account set DATA_RETENTION_TIME_IN_DAYS = 3 ;
```

```
alter account set MIN_DATA_RETENTION_TIME_IN_DAYS = 2 ;
show tables ;
create table test_client1 as select * from training.demo.client ;
show tables ; -- retetnion time is 3 ..taken from account


-- change retention at DB level
alter database set DATA_RETENTION_TIME_IN_DAYS = 5 ;
show parameters like '%retention%' in account; -- retention time is 3
show parameters like '%retention%' in database; -- overrides account, since modified at DB and flows
to schema as there is no override at schema
show parameters like '%retention%' in schema; -- gets from database
show tables; -- shows 5 overrides account
create table test_client2 as select * from training.demo.client ;
show tables; -- 5 for new table as well, 5 for all

-- Set at schema level
alter schema set DATA_RETENTION_TIME_IN_DAYS = 6 ;
show parameters like '%retention%' in account;  -- 3
show parameters like '%retention%' in database; -- 5
show parameters like '%retention%' in schema; -- 6 -- schema override will flow to tables
show tables; -- retention time is 6 for new and existing

-- setting at table level
alter table test_client1 set DATA_RETENTION_TIME_IN_DAYS = 7 ;
alter table test_client2 set DATA_RETENTION_TIME_IN_DAYS = 8 ;
show tables; -- 6, 7,8

-- setting up at create table lebel
create or replace table test_client3 (id number, name varchar(30)) data_retention_time_in_days=9;
--
show tables; -- 6,7,8.9

-- Increasing retention - it increases the time period for which the data is available in time travel => 5 ->
10, does not apply to data already in fail safe
-- Decreasing retention - it reduces the time period for which the data is available in time travel => 6 ->
4. some data move to fail-safe


-- check the storage cost and different buckets

-- UI only shows Stage, Database and Fail Safe
-- Below query shows time travel bytes
-- Since time travel stores the data to be available it incurs the cost.
-- We can check the cost –bucket for time travel via UI as well -- as using SQL queries
-- check the storage cost and different buckets

select * from snowflake.account_usage.table_storage_metrics ;
```

```
-- cleanup
drop database training_rt ;
```

FAIL SAFE

1. Historical Data Protection in case of system failure or disaster
2. 7-days of historic data can be recovered via fail safe. 7 is non configurable
3. Permanent Tables - 7 days , Temporary/Transient Tables -> 0 days
4. Should be the last resort
5. Not a regular method to access the data after time travel ends
6. Only snowflake can recover the data - can take several hours to days
7. Contributes to storage cost
8. Fail Safe can not be disabled in snowflake
9. Period starts once time travel ends

Reference: Snowflake documentation
https://docs.snowflake.com/en/user-guide/data-failsafe.html

Demo:

Check via UI - Different Storage types - Stage, Database and Fail Safe

```
select * from snowflake.account_usage.storage_usage order by usage_date desc;
select * from snowflake.account_usage.table_storage_metrics order by id desc;
```

Real life example

Trades (operational) - Time travel 90 days

Time Travel : 90 Days

1	2	3	.	.	.	.	88	89	90

91th day, Fail Safe:Day1, Time Travel: Day2 to Day91

1	2	3	.	.	.	.	.	89	90	91

92th day, Fail Safe:Day1 to Day2, Time Travel: Day3 to Day92

1	2	3	.	.	.	.	.	.	90	91	92

.
.
.

97th day, Fail Safe:Day1 to Day7, Time Travel: Day8 to Day97

1	2	3	4	5	6	7	8	.	.	.	.	.	.	95	96	97

98th day, Lost Forever: Day1, Fail Safe:Day2 to Day8, Time Travel: Day9 to Day98

1	2	3	4	5	6	7	8	9	.	.	.	.	.	.	96	97	98

100th day, Lost Forever: Day1 to Day3, Fail Safe:Day4 to Day10, Time Travel: Day11 to Day100

1	2	3	4	5	6	7	8	9	10	11	.	.	.	.	.	98	99	100

Trades (operational) - Time travel 0 days

Ist day, Fail Safe:Day1, Time Travel:0

1

2nd day, Fail Safe:Day1 to Day2, Time Travel: 0

1	2

.
.
.

7th day, Fail Safe:Day1 to Day7, Time Travel: 0

1	2	3	4	5	6	7

8th day, Lost Forever: Day1, Fail Safe:Day2 to Day8, Time Travel: 0

1	2	3	4	5	6	7	8

ZERO COPY CLONING

Overview of Zero Copy Cloning:

What: Cloning as the name suggests is the copying database, schema and tables (not temporary)

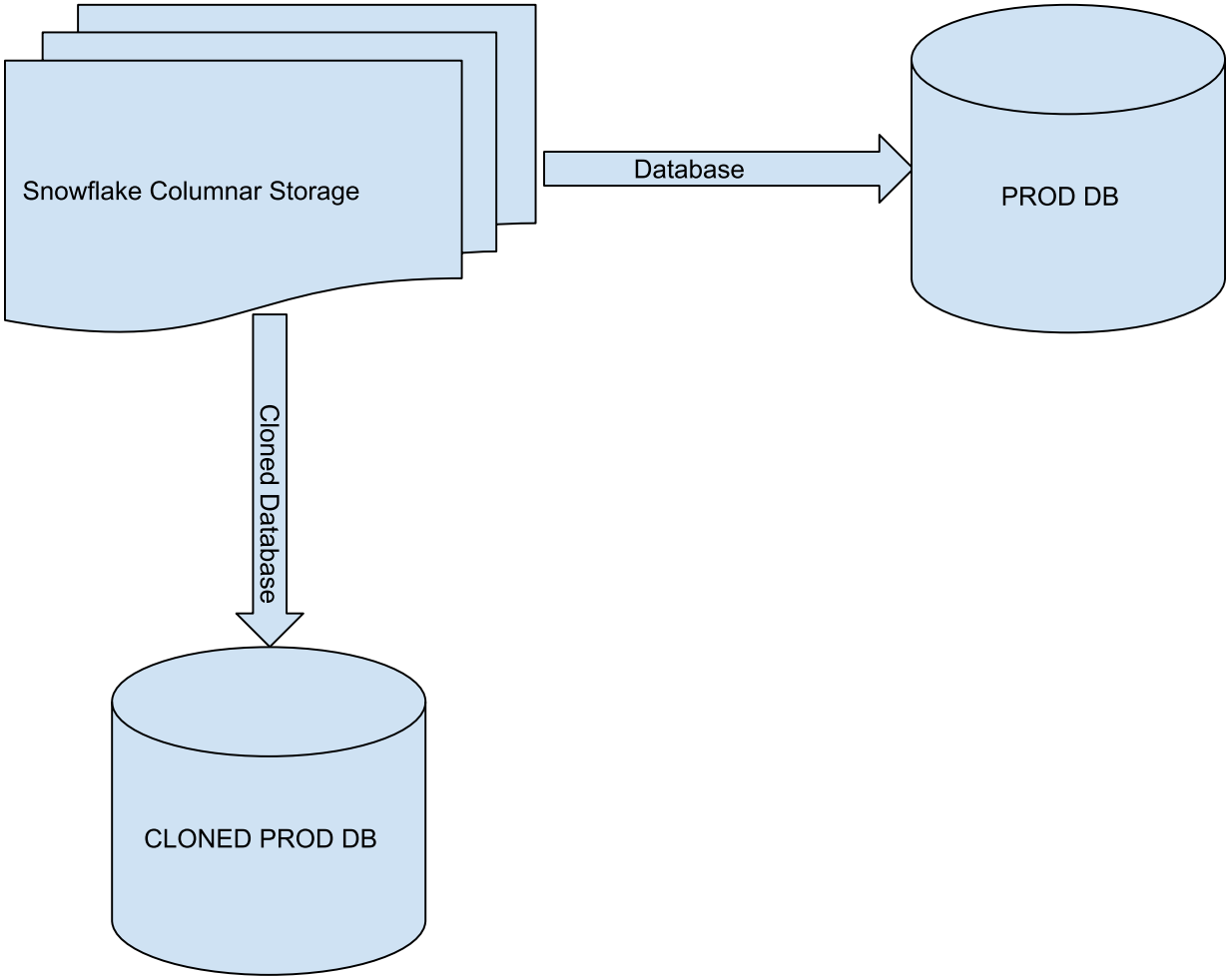
Why: Real life examples

Reproducing a production problem in a non production environment to fix it.

a. Long running queries. It may happen that a query for a batch job is running long in production but runs faster in a non-production environment. This happens due to the data volume difference across the environment. Hence copying the prod DB to Non Prod is needed to reproduce the problem.

b. Code debug and identify the issue. We can not change processing code to catch the real issue. Once we have a copy of the prod database, various placeholders can be created in the code to catch the issue and fix the issue.

Zero-Copy Cloning: (Metadata Operation)



1. It is achieved by MetaData operation
2. Data storage cost - only for the data changes. Stored in Partitions.
3. Cloned object is independent of the main object.
4. Modifications can be done on clones object without affecting the main object

How:

CLONE command takes care of everything

```
CREATE [ OR REPLACE ] { STAGE | FILE FORMAT | SEQUENCE | STREAM | TASK } [ IF NOT EXISTS ] <object_name>
  CLONE <source_object_name>
  ...
```

```
CREATE [ OR REPLACE ] { DATABASE | SCHEMA | TABLE } [ IF NOT EXISTS ] <object_name>
  CLONE <source_object_name>
        [ { AT | BEFORE } ( { TIMESTAMP => <timestamp> | OFFSET => <time_difference> | STATEMENT => <id> } ) ]
  ...
```

The metadata of the cloned object is same as original object e.g. keys, etc

Objects that can be cloned other than database, schema and tables are -

a. Stages
b. File Formats
c. Sequence
d. Stream
e. Tasks
f. Foreign key constraints

What can not be cloned.

1. External Tables
2. Internal named Stages

Demo:

```
-------------------- ZERO COPY CLONING-------------

-- PERMANENT TABLES -

-- Create table client
create or replace table client
(  id NUMBER(38,0), first_name VARCHAR(16), last_name VARCHAR(50),
   sex VARCHAR(1), ethinicity VARCHAR(30), ssn VARCHAR(15),
   street_address VARCHAR(90),status VARCHAR(10)
);


delete from client ;
insert into client values (111111, 'James',      'Schwartz', 'M', 'American','342-76-9087','5676 Washington Street','ACTIVE') ;
insert into client values (222222, 'Jessica',    'Escobar',  'F', 'Hispanic','456-93-5629','3234 WateringCan Drive','INACTIVE') ;
insert into client values (333333, 'Ben',        'Hardy',    'M', 'American','876-98-3245','6578 Historic Circle','INACTIVE') ;
insert into client values (444444, 'Anjali',     'Singh',    'F', 'Indian American','435-87-6532','8978 Autumn Day Drive','ACTIVE') ;
insert into client values (555555, 'Dean',       'Tracy',    'M', 'African','767-34-7656','2343 India Street','ACTIVE') ;

-- check the data
select * from client ; -- 5 rows

-- clone the table client
create table cln_client clone client ;

-- check the cloned table
select * from cln_client ;

-- check the tables
show tables;

-- Lets do some DML operations and validate that the original table is not impacted

-- check insert / update / delete operations effect
```

```
-- insert
insert into cln_client values (666666, 'Shawn', 'Pollock', 'M', 'American','457-34-4532','2343 Washington Street','ACTIVE');

select * from cln_client ; -- 6 rows
select * from client ; -- 5 rows (no effect on main table)

-- delete
delete from cln_client where id in (666666, 555555)  ;
select * from cln_client ; -- 4 rows
select * from client ; -- 5 rows

-- update
update cln_client set status = 'INACTIVE' where id = 111111 ;
select * from cln_client ;
select * from client ;


-- *POINT to NOTE*
-- For large table cloning which takes some time to run
-- Do not execute DML until cloning is done

-- To avoid costs some times the data retetnion is kept at 0
--1) If data is deleted from source table while cloning is in progress, the cloned object might not have the data
--2) How can this be avoided, set retention as 1, then clone once clining is done reset back to 0
--3) You can keep the retention as 0 for cloned tablles to avoid costs
--4) We learned in previous session how to set retention at table level


-- DDL also, Do it once the cloning is already done
alter table client rename to new_client ;
select * from cln_client
select * from client ;
select * from new_client ;

-- *** POINT TO NOTE ****
-- The above DDL worked fine, but we we are cloning a big table and it is running and taking some time
-- At that time if DDL (source table rename) is performed then Cloning might not work and say object now found
-- Hence refrain from DDL while cloning in Progress
```

```
-- COPY GRANTS ---
alter table client rename to new_client ;

grant select on new_client to sysadmin with grant option;

show grants on table new_client ;
show grants on table cln_client ;

-- with grants with out grants (wog)
create table cln_client_wog clone new_client  ;
show grants on table cln_client_wog ;
-- clone with grants (wg)
create table cln_client_wg clone new_client copy grants ;
show grants on table cln_client_wg ;
```

----------TEMPORARY AND EXTERNAL TABLES----------

-- Can not clone temporary and external tables. Temporary tables can be cloned as temporary or transient. Lets see

```
show tables ;

-- temporary table can not be cloned
create or replace temporary table client_tmp as select * from new_client where sex = 'M' ;
select * from client_tmp ;

-- create a clone of temporary table - does not work (This command tries to create a permanent table)
```

```
create or replace table cln_client_tmp clone client_tmp ;

-- Clone of temporary table can be cretaed to temporary and transient table
-- temporary clone of temporary table is possible
create or replace temporary table cln_client_tmp clone client_tmp ;
-- tansient clone of temp table is possible
create or replace transient table cln_client_tmp_tr clone client_tmp ;



-- external table clone not allowed
create or replace storage integration int_gcp
  type = external_stage
  storage_provider = gcs
  enabled = true
  storage_allowed_locations = ('gcs://snowflakedatafiles789/csvfiles/dataload')
  comment = 'This is the integration object for loading / unloading the files from GCP to Snowflake' ;

-- Describe Intergation Object
desc integration int_gcp ;
-- atgmtgopdp@va3-22da.iam.gserviceaccount.com


-- create file format - since it will be used in external stage and copy command. so creating is better
create or replace file format file_format_csv skip_header = 1 compression = none;

--describe file formats
describe file format file_format_csv ;

-- create external stage for loading data
create or replace stage ext_stage_ld
  url = 'gcs://snowflakedatafiles789/csvfiles/dataload'
  storage_integration = int_gcp
  file_format = file_format_csv ;

-- Show stages
show stages ;


--- create external table
create or replace external table promotions_ext
  location=@ext_stage_ld
  auto_refresh = false
  file_format  = (type=csv);


-- Check the data of external table
select metadata$filename, metadata$file_row_number, pet.* from promotions_ext pet ;
-- create clone on external table - Does not work
```

```
create table cln_promotions_ext clone promotions_ext ;
```

```
---
show tables ;
alter table if exists new_client rename to client ;
drop table if exists client_tmp ;
drop table if exists cln_client ;
drop table if exists cln_client_tmp ;
drop table if exists cln_client_tmp_tr ;
drop table if exists cln_client_wg ;
drop table if exists cln_client_wog ;
drop table if exists promotions_ext ;

drop table if exists CANDIDATES ;
drop table if exists CLIENT ;
drop table if exists CLN_CLIENT ;
drop table if exists CLIENT_RT ;
drop table if exists INCOME_BAND ;
drop table if exists INCOME_BAND_TEMP ;
drop table if exists JOB_DETAILS ;
drop table if exists RAW_APPLICANT_STAGING ;
drop table if exists TEST_CLIENT ;
```

```
----------------------clone database and schemas -----------
-- shows external table
show tables ;

-- does not clone the external tables and internal stages (user or table stages)
-- @~/vksingh - use stage, @%client - table stage

-- TRANSIENT SCHEMA -- Lets create the transient schema from snowflake sample data
-- to avoid extra storage cost due to fail safe

create or replace transient schema cln_tpcds_sf100tcl clone snowflake_sample_data.tpcds_sf100tcl ; --
error why?

create or replace transient schema cln_demo clone demo ;

use schema cln_demo ;
show tables ; -- we don't see the external tables

-- schema demo
use schema demo ;
show tables ; -- we see external table in main schema


-----------DATABASE -------
create or replace transient database cln_training clone training ;

use database cln_training ;
use schema demo ;
---
show tables ;

---- main schema
use database training ;
use schema demo ;

show tables ;

-- schema across database can be created by giving the <database>.<schema> qualifiers
-- clone of a clone
create or replace transient schema cln1_demo clone cln_training.cln_demo ;

use database training ;
use schema cln1_demo ;


-- clean up

drop database cln_training ;
```

```
drop schema cln1_demo ;
drop schema cln_demo ;

show databases ;
show schemas ;
```

SWAP:

```
ALTER TABLE [ IF EXISTS ] <name> SWAP WITH <target_table_name>
```

```
ALTER SCHEMA [ IF EXISTS ] <name> SWAP WITH <target_schema_name>
```

```
ALTER DATABASE [ IF EXISTS ] <name> SWAP WITH <target_db_name>
```

Demo:

-- SWAP -

select * from client ;

create or replace table clients as select * from client where status = 'ACTIVE' ;

select * from client ; -- 5 rows

select * from clients ; -- 2 rows

-- how can we swap both

alter table client swap with clients ;

select * from client ; -- 2 rows

select * from clients ; -- 5 rows

-- cleanup
alter table clients swap with client ;
drop table clients

```
-- SWAP Schema


-- clone a schema
create or replace schema demos clone demo ;
-- schema and see tables
use schema demos ;
show tables ;
-- drop the table client from demos
drop table client ;
show tables ; -- 3 tables
-- Now demos don't have the client and demo have it.
-- lets validate

use schema demo ;
show tables ; -- 4 tables


-- swap schemas
alter schema demos swap with demo ;

use schema demo ;
show tables ; -- 3 tables

use schema demos ;
show tables ; -- 4 tables

-- cleanup
alter schema demo swap with demos ;

use schema demo ;
show tables ;
drop schema demos ;


-- -- create or replace table client
-- (  id NUMBER(38,0), first_name VARCHAR(16), last_name VARCHAR(50),
--    sex VARCHAR(1), ethinicity VARCHAR(30), ssn VARCHAR(15),
--    street_address VARCHAR(90),status VARCHAR(10)
-- );

-- -- insert some data
-- delete from client ;
-- insert into client values (111111, 'James', 'Schwartz', 'M', 'American','342-76-9087','5676
Washington Street','ACTIVE') ;
```

```
-- insert into client values (222222, 'Jessica', 'Escobar', 'F', 'Hispanic','456-93-5629','3234 WateringCan Drive','INACTIVE') ;
-- insert into client values (333333, 'Ben', 'Hardy', 'M', 'American','876-98-3245','6578 Historic Circle','INACTIVE') ;
-- insert into client values (444444, 'Anjali', 'Singh', 'F', 'Indian American','435-87-6532','8978 Autumn Day Drive','ACTIVE') ;
-- insert into client values (555555, 'Dean', 'Tracy', 'M', 'African','767-34-7656','2343 India Street','ACTIVE') ;

-- select * from client ;

--- SWAP Database



-- clone a database
show databases ;
create or replace database trainings clone training ;
-- schema and see tables
use database trainings ;
use schema demo ;
show tables ;
-- drop the table client from database
drop table client ;
drop table income_band ;
show tables ; -- 2 tables


-- swap databases
alter database trainings swap with training ;

use database trainings ;
use schema demo ;
show tables ; -- 4 tables

use database training ;
use schema demo ;
show tables ; -- 2 tables

-- cleanup
alter database training swap with trainings ;
show databases ;

use database training;
use schema demo ;
show tables ;

drop database trainings ;
```

--- CLONING USING TIME TRAVEL --

--- CLONING USING TIME TRAVEL --

```
show tables ;


select * from client ;
-- delete the data and copy the query ID
delete from client ;
--<queryid>
-- <01a7b8ac-0004-2eeb-0026-d9870009801e>

-- check the data
select * from client ; -- no data

-- create clone using before statement
create table cln_tt_client1 clone client
  before(statement => '01a7b8ac-0004-2eeb-0026-d9870009801e');
-- Check the First clone using before statement
select * from cln_tt_client1 ;

-- create clone using offset
create table cln_tt_client2 clone client at(offset => -600); -- 10 minutes before state of table
-- check the second clone using offset
select * from cln_tt_client2 ;


-- cleanup
insert into client select * from cln_tt_client1 ;
drop table if exists cln_tt_client1 ;
drop table if exists cln_tt_client2 ;
select * from client ;
```

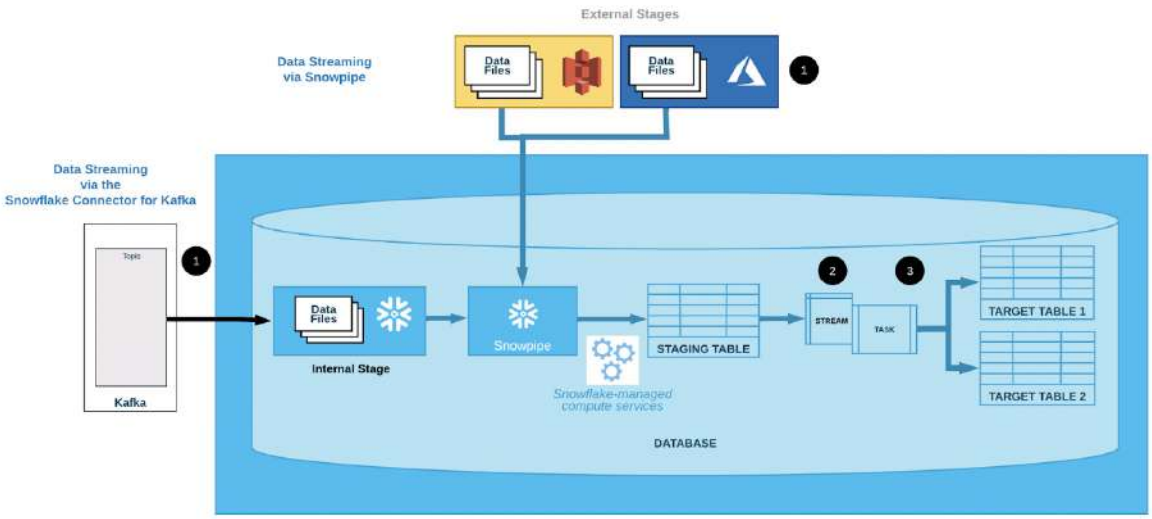
DATA SHARING VERSUS CLONING

	Data Sharing	Cloning
Data Consistency	Automatic	Static point in time copy
Cost	No additional cost	Additional cost if data is modified
Readonly	Yes	No
Location	Data can be shared externally to consumer and reader	Data stays locally in the account.

STREAMS AND TASKS

Continuous Data Pipeline:

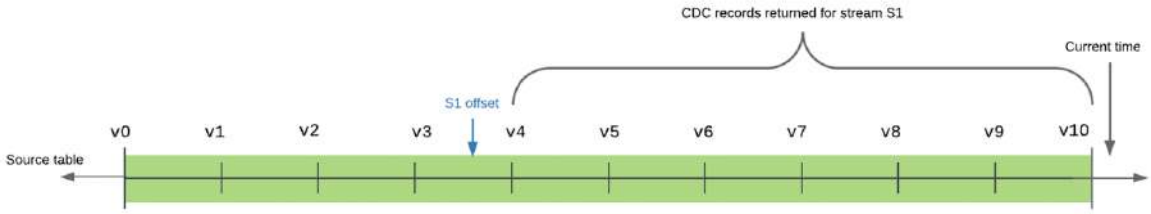
- Continuous data load via Snowpipe
- Change data capture via Streams
- Schedule recurring activities via Tasks



References: https://docs.snowflake.com/en/user-guide/data-pipelines-intro.html

STREAMS:

1. It is a database object. -> create or replace stream
2. Does not contain any data. Stores **Offset or a bookmark** when created.



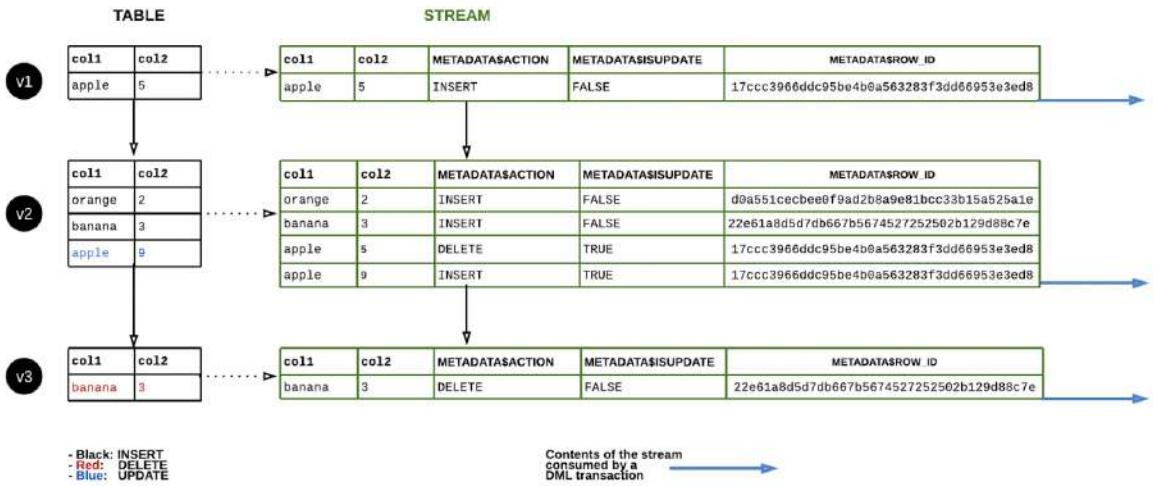
References: https://docs.snowflake.com/en/user-guide/streams-intro.html

3. Offset moves only if the stream is used in DML (truncate also) and DML is committed.
    a. Controlled by AUTOCOMMIT OR
    b. Explicit commit.
    c. To advance offset without DML
        i. Either recreate stream
        ii. OR insert the data to a temp table from stream using the false condition.

4. Stream additional columns
    a. METADATA$ACTION

b. METADATA$ISUPDATE
c. METADATA$ROW_ID

5. Data flow process



References:
https://docs.snowflake.com/en/user-guide/streams-intro.html

6. Types of Streams
   a. Standard - For tables/ view - DML (including truncates)
      i. Insert
      ii. Update
      iii. Delete
      iv. Truncate
   b. Append Only - For tables / views Captures Only inserts
      i. Insert

c. Insert Only - external tables only - if new file is added at cloud storage - stream captures
    i. Insert

7. Multiple Consumers
    a. Create streams for each consumers

8. CHANGES Clause
    a. Tables
    b. Views

9. Considerations
    a. Recreating an object makes objects's stream stale
    b. Unconsumed records are not accessible in clone (Schema or DB)
    c. Renaming an object does not make the stream invalid or stale
    d. Stream become stale when its offset is out of data retention period
    e. For tables having retention less than 14 days, unconsumed streams are extended to not go stale for the extension time.Hence Cost is affected.

DATA_RETENTION_TIME_IN_DAYS	MAX_DATA_EXTENSION_TIME_IN_DAYS	Consume Streams in X Days
14	0	14
1	14	14
0	90	90

References: https://docs.snowflake.com/en/user-guide/streams-intro.html

Demo:

```
-- Streams --

-- Create a table to demonstrate
create or replace table raw_applicant_staging
(  id number,
   first_name varchar,
   last_name varchar,
   sex varchar,
   ethinicity varchar,
```

```
  ssn varchar,
  street_address varchar,
  education_level varchar,
  years_of_experience number,
  job_id number
);


select * from raw_applicant_staging ;


-- job_details --

create or replace table job_details (
   job_id number,
   name varchar,
   city varchar,
   state varchar,
   education_level varchar
   ) ;

delete from job_details ;
insert into job_details values (10, 'Painter', 'Raleigh', 'NC','High School') ;
insert into job_details values (20, 'Software Engineer', 'Raleigh', 'NC','Masters') ;
insert into job_details values (30, 'Data Architect', 'Raleigh', 'NC','Undergrad') ;
insert into job_details values (40, 'Vice President', 'Raleigh', 'NC','Masters') ;
insert into job_details values (50, 'Associate', 'Raleigh', 'NC','Masters') ;

--
select * from job_details ;

-- candidates
create or replace table candidates (
   id number,
   first_name varchar,
   last_name varchar,
   sex varchar,
   ethinicity varchar,
   ssn varchar,
   street_address varchar,
   candidate_education_level varchar,
   years_of_experience number,
   job_id number,
   job_name varchar,
   job_city varchar,
   job_state varchar,
   required_education_level varchar,
   status varchar,
   comments varchar
```

```
    ) ;

-- populate the table candidates
delete from candidates ;
insert into candidates
values (100000, 'James',     'Schwartz', 'M', 'American','342-76-9087','5676 Washington Street','High
School', 5,10, 'Painter', 'Raleigh', 'NC','High School','SHORTLISTED','The education level of candidate
matched with job') ;

-- check the data
select * from candidates ;

-- create stream
create or replace stream strm_applicant1 on table raw_applicant_staging ;

drop stream strm_applicant ;
desc stream strm_applicant1 ;
-- stale_after => Post 14 days
show streams ;

show parameters like '%extension%retention%' in database ;

show parameters like '%retention%' in account; -- 1
show parameters like '%extension%' in account; -- 14

-- Lets insert some data in raw-applicant_staging table

insert into raw_applicant_staging values (111111, 'James',     'Schwartz', 'M',
'American','342-76-9087','5676 Washington Street','High School', 5,10) ;
insert into raw_applicant_staging values (222222, 'Jessica',   'Escobar',  'F',
'Hispanic','456-93-5629','3234 WateringCan Drive','Undergrad', 4, 10) ;
insert into raw_applicant_staging values (333333, 'Ben',       'Hardy',    'M',
'American','876-98-3245','6578 Historic Circle','Masters', 6, 30) ;
insert into raw_applicant_staging values (444444, 'Anjali',     'Singh',   'F', 'Indian
American','435-87-6532','8978 Autumn Day Drive','Masters', 8,20) ;
insert into raw_applicant_staging values (555555, 'Dean',      'Tracy',    'M', 'African','767-34-7656','2343
India Street','Undergrad', 2,50) ;

-- check data in staging
select * from raw_applicant_staging ;

-- check stream
select * from strm_applicant1; -- Observe extra columns

-- Pull the records which are inserted
select * from strm_applicant1 where metadata$action = 'INSERT' and metadata$isupdate = 'FALSE';

-- Update the data in staging
update raw_applicant_staging set job_id = 10 where id = 555555 ;
```

```
-- check stream
select * from raw_applicant_staging;
select * from strm_applicant1; -- Observe extra columns

-- Pull the records which are inserted
delete from raw_applicant_staging where id = 555555 ;
--
select * from strm_applicant1;

-- off set is not getting reset.since we are not consuming..

--------------------------------------------------------------------------------
-- lets delete the data from staging..and see consumption of stream
delete from raw_applicant_staging ;
--staging cleaned
select * from raw_applicant_staging ;
-- stream no records
select * from strm_applicant1;

-- INSERT Staging
insert into raw_applicant_staging values (111111, 'James',     'Schwartz', 'M',
'American','342-76-9087','5676 Washington Street','High School', 5,10) ;
insert into raw_applicant_staging values (222222, 'Jessica',   'Escobar',  'F',
'Hispanic','456-93-5629','3234 WateringCan Drive','Undergrad', 4, 10) ;
insert into raw_applicant_staging values (333333, 'Ben',       'Hardy',    'M',
'American','876-98-3245','6578 Historic Circle','Masters', 6, 30) ;
insert into raw_applicant_staging values (444444, 'Anjali',     'Singh',   'F', 'Indian
American','435-87-6532','8978 Autumn Day Drive','Masters', 8,20) ;
insert into raw_applicant_staging values (555555, 'Dean',      'Tracy',    'M', 'African','767-34-7656','2343
India Street','Undergrad', 2,50) ;

--consume Stream
select * from candidates ;
-- check data of job details
select * from job_details ;


-- Consume streams -- starting with insert
merge into candidates tgt
using
(select
   id ,
   first_name ,
   last_name ,
   sex ,
   ethinicity ,
   ssn ,
   street_address ,
```

```
    str.education_level as ed_level,
    years_of_experience ,
    jdtl.job_id ,
    name ,
    city ,
    state ,
    jdtl.education_level as jreq_level ,
    case when str.education_level = jdtl.education_level then 'SHORTLISTED ' else 'STAGED' end as
status,
    case when str.education_level = jdtl.education_level then 'The education level of candidate matched
with job'
                                                else 'Application received from candidate'
    end  as comments,
    str.metadata$action,
    str.metadata$isupdate
   from strm_applicant1 str inner join job_details jdtl on (str.job_id = jdtl.job_id)
 ) src
 on tgt.id = src.id
 -- insert clause
 when not matched and src.metadata$action = 'INSERT' and metadata$isupdate = 'FALSE'
 then insert
 values (
    id ,
    first_name ,
    last_name ,
    sex ,
    ethinicity ,
    ssn ,
    street_address,
    ed_level,
    years_of_experience ,
    job_id ,
    name ,
    city ,
    state ,
    jreq_level,
    status,
    comments
 )
 ;

-- check
select * from candidates ;
-- check if stream is consumed
select * from strm_applicant1 ;

--- Update Operation on raw
update raw_applicant_staging set job_id = 10 where id = 555555 ;
select * from strm_applicant1 ;
```

```
-- update education level in second update
update raw_applicant_staging set education_level = 'High School' where id = 555555 ;
select * from strm_applicant1 ;
-- Consume streams -- update
merge into candidates tgt
using
(select
    id ,
    first_name ,
    last_name ,
    sex ,
    ethinicity ,
    ssn ,
    street_address ,
    str.education_level as ed_level,
    years_of_experience ,
    jdtl.job_id ,
    name ,
    city ,
    state ,
    jdtl.education_level as jreq_level ,
    case when str.education_level = jdtl.education_level then 'SHORTLISTED ' else 'STAGED' end as
status,
    case when str.education_level = jdtl.education_level then 'The education level of candidate matched
with job'
                                        else 'Application recived from candidate'
    end  as comments,
    str.metadata$action,
    str.metadata$isupdate
   from strm_applicant1 str inner join job_details jdtl on (str.job_id = jdtl.job_id)
 ) src
 on tgt.id = src.id
 -- insert clause
 when not matched and src.metadata$action = 'INSERT' and metadata$isupdate = 'FALSE'
 then insert
 values (
    id ,
    first_name ,
    last_name ,
    sex ,
    ethinicity ,
    ssn ,
    street_address,
    ed_level,
    years_of_experience ,
    job_id ,
    name ,
    city ,
    state ,
```

```
   jreq_level,
   status,
   comments
 )
 --update
 when matched and src.metadata$action = 'INSERT' and metadata$isupdate = 'TRUE'
 then update
   set tgt.job_id = src.job_id,
       tgt.candidate_education_level = src.ed_level,
       tgt.required_education_level = src.jreq_level,
       tgt.status = src.status,
       tgt.comments = src.comments
 ;
-- check target table
select * from candidates ;


--- DELETE
delete from raw_applicant_staging where id = 555555 ;
select * from strm_applicant1 ;


-- Consume
-- Consume streams -- update
merge into candidates tgt
using
(select
   id ,
   first_name ,
   last_name ,
   sex ,
   ethinicity ,
   ssn ,
   street_address ,
   str.education_level as ed_level,
   years_of_experience ,
   jdtl.job_id ,
   name ,
   city ,
   state ,
   jdtl.education_level as jreq_level ,
   case when str.education_level = jdtl.education_level then 'SHORTLISTED ' else 'STAGED' end as
status,
   case when str.education_level = jdtl.education_level then 'The education level of candidate matched
with job'
                                        else 'Application recived from candidate'
   end  as comments,
   str.metadata$action,
   str.metadata$isupdate
```

```
  from strm_applicant1 str inner join job_details jdtl on (str.job_id = jdtl.job_id)
 ) src
 on tgt.id = src.id
 -- insert clause
 when not matched and src.metadata$action = 'INSERT' and metadata$isupdate = 'FALSE'
 then insert
 values (
   id ,
   first_name ,
   last_name ,
   sex ,
   ethinicity ,
   ssn ,
   street_address,
   ed_level,
   years_of_experience ,
   job_id ,
   name ,
   city ,
   state ,
   jreq_level,
   status,
   comments
 )
 --update
 when matched and src.metadata$action = 'INSERT' and metadata$isupdate = 'TRUE'
 then update
   set tgt.job_id = src.job_id,
       tgt.candidate_education_level = src.ed_level,
       tgt.required_education_level = src.jreq_level,
       tgt.status = src.status,
       tgt.comments = src.comments
 -- delete
 when matched and src.metadata$action = 'DELETE' and metadata$isupdate = 'FALSE'
 then delete
 ;
-- check target table
select * from candidates ;

--- Finally we can run the above statement in a frequency to take care of any insert / update / delete



-- Types of Streams
-- Standard, Append Only
create or replace table raw_applicant_staging
(  id number,
   first_name varchar,
   last_name varchar,
```

```
  sex varchar,
  ethinicity varchar,
  ssn varchar,
  street_address varchar,
  education_level varchar,
  years_of_experience number,
  job_id number
);

select * from raw_applicant_staging ;

-- Default Stream
create or replace stream strm_applicant1 on table raw_applicant_staging ;
-- Append Only Stream
create or replace stream strm_applicant2 on table raw_applicant_staging append_only = true ;
--describe default stream
desc stream strm_applicant1 ;
desc stream strm_applicant2 ;

--show all streams
show streams ;

-- Lets insert some data in raw-applicant_staging table

insert into raw_applicant_staging values (111111, 'James',     'Schwartz', 'M',
'American','342-76-9087','5676 Washington Street','High School', 5,10) ;
insert into raw_applicant_staging values (222222, 'Jessica',   'Escobar',  'F',
'Hispanic','456-93-5629','3234 WateringCan Drive','Undergrad', 4, 10) ;
insert into raw_applicant_staging values (333333, 'Ben',       'Hardy',    'M',
'American','876-98-3245','6578 Historic Circle','Masters', 6, 30) ;
insert into raw_applicant_staging values (444444, 'Anjali',     'Singh',   'F', 'Indian
American','435-87-6532','8978 Autumn Day Drive','Masters', 8,20) ;
insert into raw_applicant_staging values (555555, 'Dean',      'Tracy',    'M', 'African','767-34-7656','2343
India Street','Undergrad', 2,50) ;

--check data in streams
select * from strm_applicant1 ;
select * from strm_applicant2 ;

-- delete the data in staging table
delete from raw_applicant_staging where id in (111111,222222,333333) ;

-- check data
select * from strm_applicant1 ; -- shows 2 rows
select * from strm_applicant2 ; -- shows 5 rows - does not capture the delete

--update the data
update raw_applicant_staging set education_level = 'High School' where id = 555555 ;
```

```
-- check data
select * from strm_applicant1 ; -- shows 2 rows, and id 555555 updated the education level to High School
select * from strm_applicant2 ; -- shows 5 rows - does not capture the delete, no impact of update

-- truncate the data
truncate table raw_applicant_staging ;

-- check data
select * from strm_applicant1 ; -- No rows, reflects the truncate
select * from strm_applicant2 ; -- shows 5 rows - No effect of truncate

-- NET EFFECT of Default Stream,
insert into raw_applicant_staging values (666666, 'Sam',      'Hill',    'M', 'American','767-84-7346','2343 Historic Street','Masters', 2,50) ;
delete from raw_applicant_staging where id = 666666 ;
select * from raw_applicant_staging ;

-- check data
select * from strm_applicant1 ; -- Insert and then delete is not reflected in the default stream. Since default stream provides the row level delta.
select * from strm_applicant2 ; -- The new row inserted is reflected.

-- How to advance offset in a stream using a false condition
create or replace table raw_applicant_temp as select * from raw_applicant_staging where 1 = 2;

-- consume stream using false cindition
insert into raw_applicant_staging values (666666, 'Sam',      'Hill',    'M', 'American','767-84-7346','2343 Historic Street','Masters', 2,50) ;

insert into raw_applicant_temp
select id ,
   first_name ,
   last_name ,
   sex ,
   ethinicity ,
   ssn ,
   street_address ,
   education_level,
   years_of_experience ,
   job_id
from strm_applicant1 where 1 = 2;

-- stream is consumed
select * from strm_applicant1 ;

-- can be consumed using a create temp table
insert into raw_applicant_staging values (666666, 'Sam',      'Hill',    'M', 'American','767-84-7346','2343 Historic Street','Masters', 2,50) ;
```

```
create or replace table raw_applicant_temp2 as
select id ,
   first_name ,
   last_name ,
   sex ,
   ethinicity ,
   ssn ,
   street_address ,
   education_level,
   years_of_experience ,
   job_id
from strm_applicant1 where 1 = 2

-- stream is consumed
select * from strm_applicant1 ;
select * from raw_applicant_temp2 ;


---------------------------------------
-- Insert Only - For external Tables

-- Go to gcs and place 2 files - promotions1 and promotions2

create or replace storage integration int_gcp
  type = external_stage
  storage_provider = gcs
  enabled = true
  storage_allowed_locations = ('gcs://snowflakedatafiles789/csvfiles/dataload')
  comment = 'This is the integration object for loading / unloading the files from GCP to Snowflake' ;

-- Describe Intergation Object
desc integration int_gcp ;
-- goyexadihx@va3-22da.iam.gserviceaccount.com


-- create file format - since it will be used in external stage and copy command. so creating is better
create or replace file format file_format_csv skip_header = 1 compression = none;

--describe file formats
describe file format file_format_csv ;

-- create external stage for loading data
create or replace stage ext_stage_ld
  url = 'gcs://snowflakedatafiles789/csvfiles/dataload'
  storage_integration = int_gcp
  file_format = file_format_csv ;

-- Show stages
```

```
show stages ;

-- list
list @ext_stage_ld ;


-- external table
create or replace external table promotions_ext
  location=@ext_stage_ld
  auto_refresh = false
  file_format  = (type=csv);


-- show
show external tables ;
desc table promotions_ext ;

-- check data in stage
select $1, $2, $3, $4 from @ext_stage_ld;

-- Check the data of external table
select metadata$filename, metadata$file_row_number, pet.* from promotions_ext pet ;

-- create insert only stream
-- Append Only Stream
create or replace stream strm_applicant3 on external table promotions_ext insert_only = true ; -- check
the word externa and insert only

-- check the data in stream
select * from strm_applicant3 ;

-- Lets add file promotions3.csv at gcs location
alter external table promotions_ext refresh ;  -- The table can be created as

--check the data in stream
select * from strm_applicant3 ;

-- delete the files from cloud location
-- refresh
alter external table promotions_ext refresh ;

--check the data in stream
select * from strm_applicant3 ;

-- add files and refresh
select metadata$filename, metadata$file_row_number, pet.* from promotions_ext pet ;
alter external table promotions_ext refresh ;
select * from strm_applicant3 ;
```

—----------

```
-- Multiple consumers

create or replace table raw_applicant_staging
(  id number,
   first_name varchar,
   last_name varchar,
   sex varchar,
   ethinicity varchar,
   ssn varchar,
   street_address varchar,
   education_level varchar,
   years_of_experience number,
   job_id number
);


-- Lets insert some data in raw-applicant_staging table

insert into raw_applicant_staging values (111111, 'James',     'Schwartz', 'M',
'American','342-76-9087','5676 Washington Street','High School', 5,10) ;
insert into raw_applicant_staging values (222222, 'Jessica',   'Escobar',  'F',
'Hispanic','456-93-5629','3234 WateringCan Drive','Undergrad', 4, 10) ;
insert into raw_applicant_staging values (333333, 'Ben',       'Hardy',    'M',
'American','876-98-3245','6578 Historic Circle','Masters', 6, 30) ;
insert into raw_applicant_staging values (444444, 'Anjali',     'Singh',   'F', 'Indian
American','435-87-6532','8978 Autumn Day Drive','Masters', 8,20) ;
insert into raw_applicant_staging values (555555, 'Dean',      'Tracy',    'M', 'African','767-34-7656','2343
India Street','Undergrad', 2,50) ;


-- adam tabe
create or replace table applicant_adam as select * from raw_applicant_staging where 1 = 2 ;
-- smith table
create or replace table applicant_smith as select * from raw_applicant_staging where 1 = 2 ;

-- select
select * from raw_applicant_staging ;

show users ;

create or replace stream strm_applicant_adam on table raw_applicant_staging ;
-- Append Only Stream
create or replace stream strm_applicant_smith on table raw_applicant_staging ;
--describe default stream
```

```
desc stream strm_applicant_adam ;
desc stream strm_applicant_smith ;

--show all streams
show streams ;

select * from strm_applicant_adam;
select * from strm_applicant_smith;

-- insert two rows..
insert into raw_applicant_staging values (222222, 'Jessica',   'Escobar',  'F',
'Hispanic','456-93-5629','3234 WateringCan Drive','Undergrad', 4, 10) ;
insert into raw_applicant_staging values (333333, 'Ben',       'Hardy',    'M',
'American','876-98-3245','6578 Historic Circle','Masters', 6, 30) ;
-- consume adam
insert into applicant_adam
  select  id ,
    first_name ,
    last_name ,
    sex ,
    ethinicity ,
    ssn ,
    street_address ,
    education_level,
    years_of_experience ,
    job_id  from strm_applicant_adam ;

-- consume smith
select * from strm_applicant_smith ;

-- So if I check the data in the main table it has three rows.
select * from raw_applicant_staging ;

-- Hence it is recommended to create the different streams for different users so that each can work
independently and consume the data independently.
-- So there is only one stream and another user want to consume it, the data is not available for the
next user.
-- In this case Adam consumed its stream, but data is still available in smith stream and he can
consume based on his requirements


-------------------------------------------------------------

-- Another feature is Change clause, where data persists even after consumption we will check that in
next session


-- Changes Clause
```

```
create or replace table raw_applicant_staging
(  id number,
   first_name varchar,
   last_name varchar,
   sex varchar,
   ethinicity varchar,
   ssn varchar,
   street_address varchar,
   education_level varchar,
   years_of_experience number,
   job_id number
);

insert into raw_applicant_staging values (111111, 'James',     'Schwartz', 'M',
'American','342-76-9087','5676 Washington Street','High School', 5,10) ;
insert into raw_applicant_staging values (222222, 'Jessica',   'Escobar',  'F',
'Hispanic','456-93-5629','3234 WateringCan Drive','Undergrad', 4, 10) ;
insert into raw_applicant_staging values (333333, 'Ben',       'Hardy',    'M',
'American','876-98-3245','6578 Historic Circle','Masters', 6, 30) ;

-- view
create or replace view vw_raw_applicant_staging as select * from raw_applicant_staging where sex =
'M' ;

select * from raw_applicant_staging ; -- 3 rows
select * from vw_raw_applicant_staging ; -- 2 rows

-- CHANGE_TRACKING --

alter table raw_applicant_staging set change_tracking = true ;
alter view vw_raw_applicant_staging set change_tracking = true ; --

-- using change_tracking
select current_timestamp;

--2022-09-26 04:00:37.675 -0700 -- track the changes from this time onwards so this is our offset
-- check existing data
select * from raw_applicant_staging ; -- 3 rows
select * from vw_raw_applicant_staging ; -- 2 rows

--insert the data post our offset
insert into raw_applicant_staging values (444444, 'Anjali',     'Singh',   'F', 'Indian
American','435-87-6532','8978 Autumn Day Drive','Masters', 8,20) ;
insert into raw_applicant_staging values (555555, 'Dean',      'Tracy',    'M', 'African','767-34-7656','2343
India Street','Undergrad', 2,50) ;

select * from raw_applicant_staging changes (information => default)   -- information can be
append_only or insert
```

```
at (timestamp => '2022-09-26 04:00:37.675 -0700'::timestamp_tz) ;      -- use the offset

select * from vw_raw_applicant_staging changes (information => default)   -- information can be
append_only or insert
at (timestamp => '2022-09-26 04:00:37.675 -0700'::timestamp_tz) ;         -- use the offset


-- delete
select current_timestamp ;
-- 2022-09-26 04:02:37.223 -0700  -- lets test the delete post this time

delete from raw_applicant_staging where id in (111111, 222222) ; -- 1 M and 1 F

select * from raw_applicant_staging changes (information => default)   -- shows 2 records deleted both
M and F
at (timestamp => '2022-09-26 04:02:37.223 -0700'::timestamp_tz) ;

select * from vw_raw_applicant_staging changes (information => default)   -- shows 1 M record
at (timestamp => '2022-09-26 04:02:37.223 -0700'::timestamp_tz) ;

--update
select current_timestamp ;
-- 2022-09-26 04:04:02.450 -0700  -- this is our update time stamp offset, we will do update post this
time.

select * from raw_applicant_staging ;
select * from vw_raw_applicant_staging ;

update raw_applicant_staging set education_level = 'High School' where id in (333333, 444444) ;

select * from raw_applicant_staging changes (information => default)   -- gives 2 delete and insert
at (timestamp => '2022-09-26 04:04:02.450 -0700'::timestamp_tz) ; -- use timestamp taken before
update

select * from vw_raw_applicant_staging changes (information => default)  -- gives 1 delete and insert
at (timestamp => '2022-09-26 04:04:02.450 -0700'::timestamp_tz) ; -- use timestamp taken before
update


-- consume the change clause and offset is not advanced
create or replace table applicant_temp as select * from raw_applicant_staging where 1 = 2;

insert into applicant_temp
   select  id ,
   first_name ,
   last_name ,
   sex ,
   ethinicity ,
   ssn ,
```

```
    street_address ,
    education_level,
    years_of_experience ,
    job_id  from raw_applicant_staging changes (information => default)
at (timestamp => '2022-09-26 04:04:02.450 -0700'::timestamp_tz) where metadata$action = 'INSERT' ;


-- The change track is not consumer OR the offset is not moved by the consumption

-- can be used in create table as well

 create table t1 as select  id ,
    first_name ,
    last_name ,
    sex ,
    ethinicity ,
    ssn ,
    street_address ,
    education_level,
    years_of_experience ,
    job_id  from raw_applicant_staging changes (information => default)
at (timestamp => '2022-09-26 04:04:02.450 -0700'::timestamp_tz) where metadata$action = 'INSERT' ;

select * from t1 ;

-----



-------------------------------------------------------------------------------------------

-- Streams considerations (d, e)

-- Stream become stale when its offset is out of data retention period
-- For tables having retention less than 14 days, unconsumed streams are extended to not go stale for
the extension time.Hence Cost is affected.

-- stale_after = current_time + max (extension, retention), retention = (max(min_retention, retention))

show streams ;
drop table raw_applicant_staging ;

alter account set DATA_RETENTION_TIME_IN_DAYS = 1 ;
alter account set MIN_DATA_RETENTION_TIME_IN_DAYS = 0 ;
alter account set MAX_DATA_EXTENSION_TIME_IN_DAYS = 14 ; --

show parameters like '%retention%' in account; -- 1, 0
show parameters like '%extension%' in account; -- 14
```

```
create or replace table raw_applicant_staging
(  id number,
   first_name varchar,
   last_name varchar,
   sex varchar,
   ethinicity varchar,
   ssn varchar,
   street_address varchar,
   education_level varchar,
   years_of_experience number,
   job_id number
);

insert into raw_applicant_staging values (111111, 'James',     'Schwartz', 'M',
'American','342-76-9087','5676 Washington Street','High School', 5,10) ;
insert into raw_applicant_staging values (222222, 'Jessica',   'Escobar',  'F',
'Hispanic','456-93-5629','3234 WateringCan Drive','Undergrad', 4, 10) ;
insert into raw_applicant_staging values (333333, 'Ben',       'Hardy',    'M',
'American','876-98-3245','6578 Historic Circle','Masters', 6, 30) ;
insert into raw_applicant_staging values (444444, 'Anjali',     'Singh',   'F', 'Indian
American','435-87-6532','8978 Autumn Day Drive','Masters', 8,20) ;
insert into raw_applicant_staging values (555555, 'Dean',      'Tracy',    'M', 'African','767-34-7656','2343
India Street','Undergrad', 2,50) ;

select * from raw_applicant_staging ;

-- create stream
create or replace stream strm_applicant4 on table raw_applicant_staging ;

desc stream strm_applicant4 ;
-- stale_after => Post 14 days

alter account set DATA_RETENTION_TIME_IN_DAYS = 2 ;
alter account set MAX_DATA_EXTENSION_TIME_IN_DAYS = 20 ; -- controls the stream stale_after

show parameters like '%retention%' in account; -- 2
show parameters like '%retention%' in database; -- 5
show parameters like '%retention%' in schema; -- 1
show parameters like '%extension%' in account; -- 20

--
desc stream strm_applicant4 ; -- stale after is 20 days later

-- making extension as 0
alter schema set data_retention_time_in_days = 8;
alter account set min_data_retention_time_in_days = 8 ;
alter account set MAX_DATA_EXTENSION_TIME_IN_DAYS = 0 ; --if this is 0 then retention period is
used for stale_after
```

```
desc stream strm_applicant4 ; -- stale after is 8 days later only considering the retention since
extension is 0


---- recreate the table makes stream stale
desc stream strm_applicant4 ;

create or replace table raw_applicant_staging
(  id number,
   first_name varchar,
   last_name varchar,
   sex varchar,
   ethinicity varchar,
   ssn varchar,
   street_address varchar,
   education_level varchar,
   years_of_experience number,
   job_id number
);

insert into raw_applicant_staging values (111111, 'James',     'Schwartz', 'M',
'American','342-76-9087','5676 Washington Street','High School', 5,10) ;
insert into raw_applicant_staging values (222222, 'Jessica',   'Escobar',  'F',
'Hispanic','456-93-5629','3234 WateringCan Drive','Undergrad', 4, 10) ;
insert into raw_applicant_staging values (333333, 'Ben',       'Hardy',    'M',
'American','876-98-3245','6578 Historic Circle','Masters', 6, 30) ;
insert into raw_applicant_staging values (444444, 'Anjali',     'Singh',   'F', 'Indian
American','435-87-6532','8978 Autumn Day Drive','Masters', 8,20) ;
insert into raw_applicant_staging values (555555, 'Dean',      'Tracy',    'M', 'African','767-34-7656','2343
India Street','Undergrad', 2,50) ;

select * from raw_applicant_staging ;

--stale
desc stream strm_applicant4 ; -- stale_after => NULL, stale = True


---Renaming the object will not make the stream in valid

-- create stream
create or replace stream strm_applicant5 on table raw_applicant_staging ;
desc stream strm_applicant5 ;

alter table raw_applicant_staging rename to raw_applicant_staging_new ;

select * from raw_applicant_staging_new ;

select * from raw_applicant_staging ; -- not exists
```

```
desc stream strm_applicant5 ;

-----unconsumed records are not available in clone


create or replace table raw_applicant_staging
(  id number,
   first_name varchar,
   last_name varchar,
   sex varchar,
   ethinicity varchar,
   ssn varchar,
   street_address varchar,
   education_level varchar,
   years_of_experience number,
   job_id number
);



select * from raw_applicant_staging ;
-- create stream
create or replace stream strm_applicant6 on table raw_applicant_staging ;
--describe stream
desc stream strm_applicant6 ; -- stale_after => NULL, stale = True

insert into raw_applicant_staging values (111111, 'James',     'Schwartz', 'M',
'American','342-76-9087','5676 Washington Street','High School', 5,10) ;
insert into raw_applicant_staging values (222222, 'Jessica',   'Escobar',  'F',
'Hispanic','456-93-5629','3234 WateringCan Drive','Undergrad', 4, 10) ;
insert into raw_applicant_staging values (333333, 'Ben',       'Hardy',    'M',
'American','876-98-3245','6578 Historic Circle','Masters', 6, 30) ;
insert into raw_applicant_staging values (444444, 'Anjali',     'Singh',   'F', 'Indian
American','435-87-6532','8978 Autumn Day Drive','Masters', 8,20) ;
insert into raw_applicant_staging values (555555, 'Dean',      'Tracy',    'M', 'African','767-34-7656','2343
India Street','Undergrad', 2,50) ;

-- select from streams
select * from strm_applicant6 ;

-- create clone schema
create or replace schema cln_demo_1 clone demo ;

-- use schema
use schema cln_demo_1 ;
-- stream in clone schema
select * from strm_applicant6 ; -- not available in clone

-- set previous schema
```

use schema demo ;
-- available in main schema
select * from strm_applicant6 ; -- not available in clone

--- We have seen streams in details, not lets understand about tasks and there we will see how can we integrate tasks with streams.

TASKS

Executes

- SQL
- Stored Procedures
- Can be combined with streams
- Can be scheduled

Tasks can be created using

```
CREATE [ OR REPLACE ] TASK [ IF NOT EXISTS ] <name>
  [ { WAREHOUSE = <string> } | { USER_TASK_MANAGED_INITIAL_WAREHOUSE_SIZE = <string> } ]
  [ SCHEDULE = '{ <num> MINUTE | USING CRON <expr> <time_zone> }' ]
  [ ALLOW_OVERLAPPING_EXECUTION = TRUE | FALSE ]
  [ <session_parameter> = <value> [ , <session_parameter> = <value> ... ] ]
  [ USER_TASK_TIMEOUT_MS = <num> ]
  [ SUSPEND_TASK_AFTER_NUM_FAILURES = <num> ]
  [ ERROR_INTEGRATION = <integration_name> ]
  [ COPY GRANTS ]
  [ COMMENT = '<string_literal>' ]
  [ AFTER <string> [ , <string> , ... ] ]
[ WHEN <boolean_expr> ]
AS
  <sql>
```

Reference:https://docs.snowflake.com/en/sql-reference/sql/create-task.html

WAREHOUSE is optional. If given it is a user-managed task and users have to provide the size as well.

If WAREHOUSE is not given then it will be a serverless task, OR snowflake managed task.

In other words the compute model for tasks can be user managed OR snowflake managed.

Demo: Creating a Task

--- We have seen streams in details, not lets understand about tasks and there we will see how can we integrate tasks with streams in the end

– CREATE TASK, SHOW TASK, EXECUTE TASK

```
-- create a task, show task, execute tasks

-- Let's create table --
create or replace table raw_applicant_staging
(  id number autoincrement start = 111111 increment  = 111111,
   first_name varchar,
   last_name varchar,
   sex varchar,
   ethinicity varchar,
   ssn varchar,
   street_address varchar,
   education_level varchar,
   years_of_experience number,
   job_id number
);
```

-- **CREATE A TASK WITHOUT SCHEDULE WITH WAREHOUSE -- USER MANAGED TASK**

```
create or replace task tsk_add_applicant
warehouse = compute_wh  -- optional, we are creating a task which uses the user defined compute
power. Hence gave the warehouse
as insert into raw_applicant_staging
(first_name, last_name,   sex,  ethinicity,  ssn,          street_address,
education_level,years_of_experience,job_id) values
('James',     'Schwartz', 'M', 'American',  '342-76-9087','5676 Washington Street','High School', 5,
10) ;

-- check task
show tasks ;

-- since there is no schedule we need to execute task
```

```
execute task tsk_add_applicant ; -- can only execute standalone task or a root task

-- select
select * from raw_applicant_staging ;
```

--- **CREATE SCHEDULED TASK WITHOUT WAREHOUSE -- A SERVERLESS TASK**

```
delete from raw_applicant_staging ;

create or replace task tsk_add_applicant_evry_min_server_less
--warehouse = compute_wh  -- No warehouse
schedule = '1 MINUTE'
as insert into
raw_applicant_staging
(first_name, last_name, sex, ethinicity,  ssn,          street_address,
education_level,years_of_experience,job_id) values
('James',     'Schwartz', 'M', 'American','342-76-9087','5676 Washington Street','High School', 5,10) ;

-- show tasks
show tasks ;

-- resume and suspend
alter task tsk_add_applicant_evry_min_server_less resume ;

-- check the data
select * from raw_applicant_staging ;

--suspend it, once we see the data
alter task tsk_add_applicant_evry_min_server_less suspend ;
```

-- CREATE SCHEDULED TASK WITH WAREHOUSE--

```
delete from raw_applicant_staging ;

create or replace task tsk_add_applicant_evry_min_usr_mnged
warehouse = compute_wh  --
schedule = '1 MINUTE'
```

```
as insert into
raw_applicant_staging (first_name, last_name, sex, ethinicity,  ssn,          street_address,
education_level,years_of_experience,job_id)
values                          ('James',     'Schwartz', 'M', 'American','342-76-9087','5676 Washington
Street','High School', 5,10) ;

-- show tasks
show tasks ;

-- resume and suspend
alter task tsk_add_applicant_evry_min_usr_mnged resume ;


-- check the data
select * from raw_applicant_staging ;
-- if we wait for 2 more minutes we will see more rows inserted

--suspend it, once we see the data
alter task tsk_add_applicant_evry_min_usr_mnged suspend ;

-- cleanup
show tables ;
drop table if exists RAW_APPLICANT_STAGING ;
show tasks ;

drop task if exists TSK_ADD_APPLICANT ;
drop task if exists TSK_ADD_APPLICANT_EVRY_MIN;
drop task if exists TSK_ADD_APPLICANT_EVRY_MIN_SERVER_LESS;
drop task if exists TSK_ADD_APPLICANT_EVRY_MIN_USR_MNGED ;
```

–CREATE TASK TO CALL STORED PROCEDURE

```
-- Let's create table --
create or replace table raw_applicant_staging
(  id number autoincrement start = 111111 increment  = 111111,
   first_name varchar,
   last_name varchar,
   sex varchar,
```

```
   ethinicity varchar,
   ssn varchar,
   street_address varchar,
   education_level varchar,
   years_of_experience number,
   job_id number
);

-- check the data
select * from raw_applicant_staging ;

-- create a procedure using Snowflake Scripting code
  create or replace procedure prc_add_applicant()
  returns varchar
  language sql
  as
  $$
    begin
      insert into raw_applicant_staging
          (first_name,   last_name, sex,  ethinicity,  ssn,          street_address,          education_level,
years_of_experience, job_id)
      values      ('James',     'Schwartz', 'M', 'American','  342-76-9087', '5676 Washington Street','High
School',    5,                    10) ;

      return 'Record Inserted' ;

   end;
  $$
  ;

-- test the procedure
select * from raw_applicant_staging ;
call prc_add_applicant() ;
select * from raw_applicant_staging ; -- procedure works

-- clean up the table
delete from raw_applicant_staging ;

-- create task to call a procedure
create or replace task tsk_add_applicant_call_prc
warehouse = compute_wh
schedule = '1 MINUTE'
as call prc_add_applicant() ;

-- show
show tasks ;

-- start the task and check
alter task tsk_add_applicant_call_prc resume ;
```

```
-- check data
select * from raw_applicant_staging ;

-- suspend the task post validations
alter task tsk_add_applicant_call_prc suspend ;

-- cleanup
drop task if exists tsk_add_applicant_call_prc ;
```

–CREATE TASK BASED ON CONDITION (USING WHEN)

```
--- CREATE TASK USING WHEN --
-- WHEN <Boolean Expression>
-- Only function allowed in WHEN is : system$stream_has_data -- that we will see when we combine
the streams and tasks
-- Let's understand the True and False date conditions

-- clean up the table
delete from raw_applicant_staging ;

select getdate() ; -- today's date and time
select dateadd(day,-1,getdate()) ; -- yesterday
select dateadd(day,1,getdate()) ; -- tomorrow



-- lets create a task using when
create or replace task tsk_add_applicant_evry_min_cond_fnc
warehouse = compute_wh
schedule = '1 MINUTE'
when getdate() between dateadd(day,1,getdate()) and dateadd(day,2,getdate()) -- A true condition
as
insert into raw_applicant_staging (first_name, last_name, sex, ethinicity,  ssn,          street_address,
education_level,years_of_experience,job_id)
values                          ('James',     'Schwartz', 'M', 'American','342-76-9087','5676 Washington
Street','High School', 5,10) ;
```

```
-- Invalid expression for task condition expression. Expecting one of the following:
[SYSTEM$STREAM_HAS_DATA]
-- It can not accept any other function


-- create task using conditions false and true and not using any function
create or replace task tsk_add_applicant_evry_min_cond_false
warehouse = compute_wh  -- optional, we are cereating a task which uses the user defined compute
power. Hence gave the warehouse
schedule = '1 MINUTE'
when false -- A False condition
as insert into raw_applicant_staging (first_name, last_name, sex, ethinicity,  ssn,          street_address,
education_level,years_of_experience,job_id)
values                          ('James',     'Schwartz', 'M', 'American','342-76-9087','5676 Washington
Street','High School', 5,10) ;

-- create task with a true condition
create or replace task tsk_add_applicant_evry_min_cond_true
warehouse = compute_wh  -- optional, we are cereating a task which uses the user defined compute
power. Hence gave the warehouse
schedule = '1 MINUTE'
when true -- A true condition
as insert into raw_applicant_staging (first_name, last_name, sex, ethinicity,  ssn,          street_address,
education_level,years_of_experience,job_id)
values                          ('Jessica',     'Escobar', 'F', 'Hispanic','456-93-5629','3234 WateringCan
Drive','Undergrad', 4,10) ;

show tasks ;

select * from raw_applicant_staging ;

alter task tsk_add_applicant_evry_min_cond_false resume ;
alter task  tsk_add_applicant_evry_min_cond_true resume ;

select * from raw_applicant_staging ;

alter task tsk_add_applicant_evry_min_cond_false suspend ;
alter task  tsk_add_applicant_evry_min_cond_true suspend ;
```

```
-- cleanup
drop task if exists tsk_add_applicant_evry_min_cond_false ;
drop task if exists tsk_add_applicant_evry_min_cond_true ;
```

— TROUBLESHOOTING THE TASKS –

```
-- Now we know that Jessica is inserted but James is not getting inserted..
-- How can we check, troubleshoot the tasks...

-- Use TASK_HISTORY()
-- get specific task
select * from table(information_schema.task_history()) where lower(name) in
('tsk_add_applicant_evry_min_cond_false') order by completed_time desc;
-- get tasks in last one hour
select * from table(information_schema.task_history()) where completed_time between dateadd(hour,
-1, getdate()) and getdate() order by completed_time desc;
-- can add the conditions of query one and two to get the specific task details via time
select * from table(information_schema.task_history()) where completed_time between dateadd(hour,
-2, getdate()) and getdate()
and lower(name) in ('tsk_add_applicant_evry_min_cond_true')
order by completed_time desc;

-- show
show tasks ; -- all should be in suspended mode

-- cleanup
drop task if exists TSK_ADD_APPLICANT_EVRY_MIN_COND_FALSE ;
drop task if exists TSK_ADD_APPLICANT_EVRY_MIN_COND_TRUE ;
```

--- **MORE on SCHEDULING TASK -- USING CRON Expression**

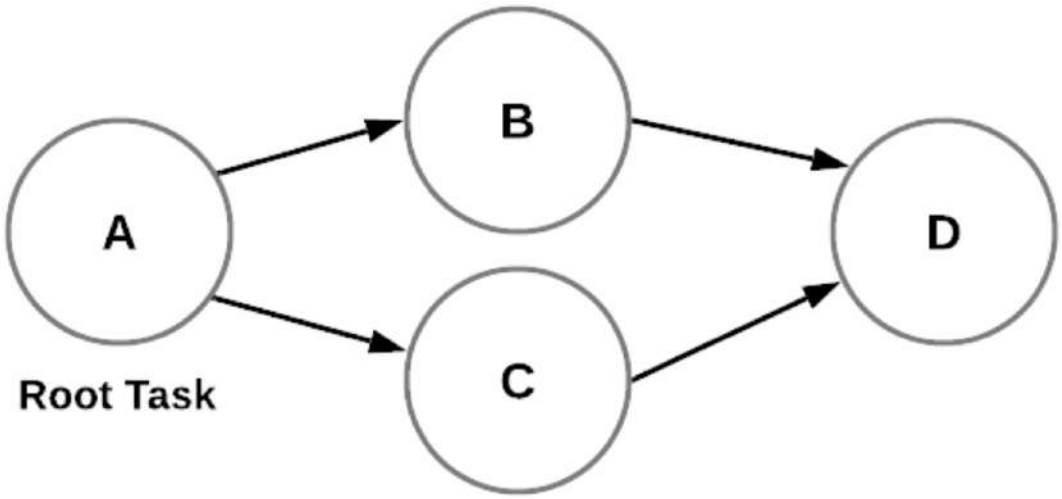
```
--- SCHEDULING TASKS USING CRON --
show parameters like '%timezone%' ;

delete from raw_applicant_staging ;

create or replace task tsk_add_applicant_cron
warehouse = compute_wh
schedule = 'USING CRON <1><2><3><4><5><6>'
as insert into raw_applicant_staging (first_name, last_name, sex, ethinicity,  ssn,  street_address,
education_level,years_of_experience,job_id)
values  ('James',  'Schwartz', 'M', 'American','342-76-9087','5676 Washington
Street','High School', 5,10) ;

-- <1> - (0-59) minute
-- <2> - (0-23) hour
-- <3> - (1-31, L) day of month
-- <4> - (1-12, JAN,FEB,MAR..DEC) month
-- <5> - (0-6,SUN,MON,TUE..SAT, L) day of week
-- <6> - (America/Los_Angeles, America/New_York, 'America/Chicago') timezone
-- EmptyValue is *

-- some examples

-- Runs at 4.50 AM every day as per LA time zone
schedule = USING CRON 50 4 * * * America/Los_Angeles

-- Runs at 4.50 AM on 2nd day of each month as per LA time zone
schedule = USING CRON 50 4 2 * * America/Los_Angeles

-- Runs at 4.50 AM on 2nd day of Oct as per LA time zone
schedule = USING CRON 50 4 2 10 * America/Los_Angeles
```

```
-- Runs at 4.50 AM in Oct every monday as per LA time zone
schedule = USING CRON 50 4 * 10 1 America/Los_Angeles

-- Runs every hour from 4:00 AM to 8:00 PM on last day of each month
schedule = USING CRON 0 4-20 L * * America/Los_Angeles

-- Runs every hour from 4:00 AM to 8:00 PM on last day of each week i.e SAT
schedule = USING CRON 0 4-20 * * L America/Los_Angeles



-- Lets see the demo.
select current_timestamp() ;

create or replace task tsk_add_applicant_cron
warehouse = compute_wh
schedule = 'USING CRON 48 11 5 10 3 America/Los_Angeles'
as insert into raw_applicant_staging (first_name, last_name, sex, ethinicity,  ssn,          street_address,
education_level,years_of_experience,job_id)
values                          ('James',     'Schwartz', 'M', 'American','342-76-9087','5676 Washington
Street','High School', 5,10) ;

-- show tasks
show tasks ;

-- resume
alter task tsk_add_applicant_cron resume ;

-- check the data
select * from raw_applicant_staging ;
--check the task
select * from table(information_schema.task_history()) where completed_time between dateadd(hour,
-1, getdate()) and getdate() order by completed_time desc;

-- Suspend
alter task tsk_add_applicant_cron suspend ;



---CONSIDERATION ABOUT DAYLIGHT SAVING -- Take care when scheduling it can have
unexpected behavior

schedule = USING CRON 0 1 * * * America/Los_Angeles -- This will run twice when time changes from
1:59:59 to 1:00:00 AM local time

schedule = USING CRON 0 2 * * * America/Los_Angeles -- This will never run when time changes from
1;59:59 to 3:00:00 AM
```

-- To avoid -> don't schedule between 1:00:00 AM and 2:00:00 AM daily OR on Sundays
-- Manually adjust twice a year during daylight saving time.

DAG (Directed Acyclic graph) OR Tree of TASK

- Tasks in a DAG are organized as per the dependencies to perform a job.
- At max there can be 1000 tasks in a DAG
- Task can have 100 children, and 100 parents at max.
- Example of a DAG



Reference: https://docs.snowflake.com/en/user-guide/tasks-intro.html

Demo: DAG

---- DAG OR TREE OF TASKS --------

```
create or replace table raw_applicant_staging
(  id number autoincrement start = 111111 increment  = 111111,
   first_name varchar,
   last_name varchar,
   sex varchar,
   ethinicity varchar,
   ssn varchar,
   street_address varchar,
   education_level varchar,
   years_of_experience number,
   job_id number
);


create or replace table job_details (
    job_id number,
    name varchar,
    city varchar,
    state varchar,
    education_level varchar
    ) ;

delete from job_details ;
insert into job_details values (10, 'Painter', 'Raleigh', 'NC','High School') ;
insert into job_details values (20, 'Software Engineer', 'Raleigh', 'NC','Masters') ;
insert into job_details values (30, 'Data Architect', 'Raleigh', 'NC','Undergrad') ;
insert into job_details values (40, 'Vice President', 'Raleigh', 'NC','Masters') ;
insert into job_details values (50, 'Associate', 'Raleigh', 'NC','Masters') ;

--
select * from raw_applicant_staging ;
select * from job_details ;


create or replace table candidates (
    id number,
    first_name varchar,
    last_name varchar,
    sex varchar,
    ethinicity varchar,
    ssn varchar,
```

```
    street_address varchar,
    candidate_education_level varchar,
    years_of_experience number,
    job_id number,
    job_name varchar,
    job_city varchar,
    job_state varchar,
    required_education_level varchar,
    status varchar,  -- status
    comments varchar,
    interview_month varchar -- extra column added
    ) ;
-- check data
select * from candidates ;

-- create the Tree

--    RootTask  (tsk_add_applicant_evry_min) -> raw applicant_staging get added
--      |
--  Child Task1 (tsk_add_candidate} -> data from raw_applicant_staging and job_details gets added to
table candidate
--                                with interview_month as unknown and status shortlisted
--      |
--  Child Task2 {tsk_update_interview_month_status} update interview month and status



-- root task
create or replace task tsk_add_applicant_evry_min
warehouse = compute_wh  -- optional, we are creating a task which uses the user defined compute
power. Hence gave the warehouse
schedule = '1 MINUTE'
as insert into raw_applicant_staging (first_name, last_name, sex, ethinicity,  ssn,          street_address,
education_level,years_of_experience,job_id)
values                          ('James',     'Schwartz', 'M', 'American','342-76-9087','5676 Washington
Street','High School', 5,10) ;

show tasks ;


-- child task 1 --
create or replace task tsk_add_candidate
warehouse = compute_wh
-- schedule = '1 MINUTE'  -- CAN NOT have a scheduler and predecssor. Hence schedule should be
removed. The schedule can be only on root OR independent task
after tsk_add_applicant_evry_min
as merge into candidates tgt
using
(select
```

```
    id ,
    first_name ,
    last_name ,
    sex ,
    ethinicity ,
    ssn ,
    street_address ,
    stg.education_level as ed_level,
    years_of_experience ,
    jdtl.job_id ,
    name ,
    city ,
    state ,
    jdtl.education_level as jreq_level ,
    case when stg.education_level = jdtl.education_level then 'SHORTLISTED' else 'STAGED' end as
status,
    case when stg.education_level = jdtl.education_level then 'The education level of candidate matched
with job'
                                                else 'Application recived from candidate'
    end  as comments
   from raw_applicant_staging stg inner join job_details jdtl on (stg.job_id = jdtl.job_id)
 ) src
 on tgt.id = src.id
 -- insert clause
 when not matched
 then insert
 values (
    id ,
    first_name ,
    last_name ,
    sex ,
    ethinicity ,
    ssn ,
    street_address,
    ed_level,
    years_of_experience ,
    job_id ,
    name ,
    city ,
    state ,
    jreq_level,
    status,
    comments,
    'Unknown'
 )
 ;

 show tasks ;
```

```
 alter task tsk_add_candidate resume;

 --
 -- Getting the month by adding a number in current date. The below query gives Oct
 select monthname(add_months(current_timestamp, 1)) as interview_month
 -- get the random number from below
 select uniform(0,11,random()) ; -- get the random number between 0 and 11
 -- random month
 select monthname(add_months(current_timestamp, uniform(0,11,random()))) as mnth



--child task 2
 create or replace task tsk_update_interview_month_status
 warehouse = compute_wh
 -- schedule = '1 MINUTE'  -- CAN NOT have a scheduler and predecssor. Hence schedule should be
removed. The schedule can be only on root OR independent task
 after tsk_add_candidate
 as update candidates set interview_month = monthname(add_months(current_timestamp,
uniform(0,11,random()))), status = 'DONE' where status = 'SHORTLISTED' ;

 show tasks ;

 alter task tsk_update_interview_month_status resume;


 -- Show Tree
 show tasks ;
 delete from raw_applicant_staging ;
 delete from candidates ;


 select * from raw_applicant_staging ;
 select * from candidates ;


-- Lets resume the Root task

alter task tsk_add_applicant_evry_min resume ;

show tasks ;

-- in few minutes we shoudl see the data flowing
 select * from raw_applicant_staging ;
 select * from candidates ;

select * from table(information_schema.task_history()) where completed_time between dateadd(hour,
-1, getdate()) and getdate() order by completed_time desc;
```

```
alter task TSK_ADD_APPLICANT_EVRY_MIN suspend ;
```

```
-- cleanup
show tasks ;
drop task if exists TSK_ADD_APPLICANT_EVRY_MIN ;
drop task if exists TSK_ADD_CANDIDATE;
drop task if exists TSK_UPDATE_INTERVIEW_MONTH_STATUS;
```

—-- TASKS WITH STREAMS —-------

```
 --------------------------------
 -- So far we have seen streams and tasks. Now let's focus on how we can really see the combination
of two.
 --  In order to fetch the data from streams into task and run in the scheduled way

 -- Create a table to demonstrate
create or replace table raw_applicant_staging
(  id number,
   first_name varchar,
   last_name varchar,
   sex varchar,
   ethinicity varchar,
   ssn varchar,
   street_address varchar,
   education_level varchar,
   years_of_experience number,
   job_id number
);
```

```
select * from raw_applicant_staging ;
-- job_details --
```

```
create or replace table job_details (
   job_id number,
   name varchar,
   city varchar,
   state varchar,
   education_level varchar
   ) ;

delete from job_details ;
insert into job_details values (10, 'Painter', 'Raleigh', 'NC','High School') ;
insert into job_details values (20, 'Software Engineer', 'Raleigh', 'NC','Masters') ;
insert into job_details values (30, 'Data Architect', 'Raleigh', 'NC','Undergrad') ;
insert into job_details values (40, 'Vice President', 'Raleigh', 'NC','Masters') ;
insert into job_details values (50, 'Associate', 'Raleigh', 'NC','Masters') ;

--select the data
select * from job_details ;

-- candidates
create or replace table candidates (
   id number,
   first_name varchar,
   last_name varchar,
   sex varchar,
   ethinicity varchar,
   ssn varchar,
   street_address varchar,
   candidate_education_level varchar,
   years_of_experience number,
   job_id number,
   job_name varchar,
   job_city varchar,
   job_state varchar,
   required_education_level varchar,
   status varchar,
   comments varchar
   ) ;

-- check the data
select * from candidates ;

-- create stream
create or replace stream strm_applicant1 on table raw_applicant_staging ;

-- create task using stream
create or replace task tsk_add_candidate_evry_min_use_strm
warehouse = compute_wh
schedule = '1 MINUTE'  -- if you don't provide schedule you have to run it manually by execute task
```

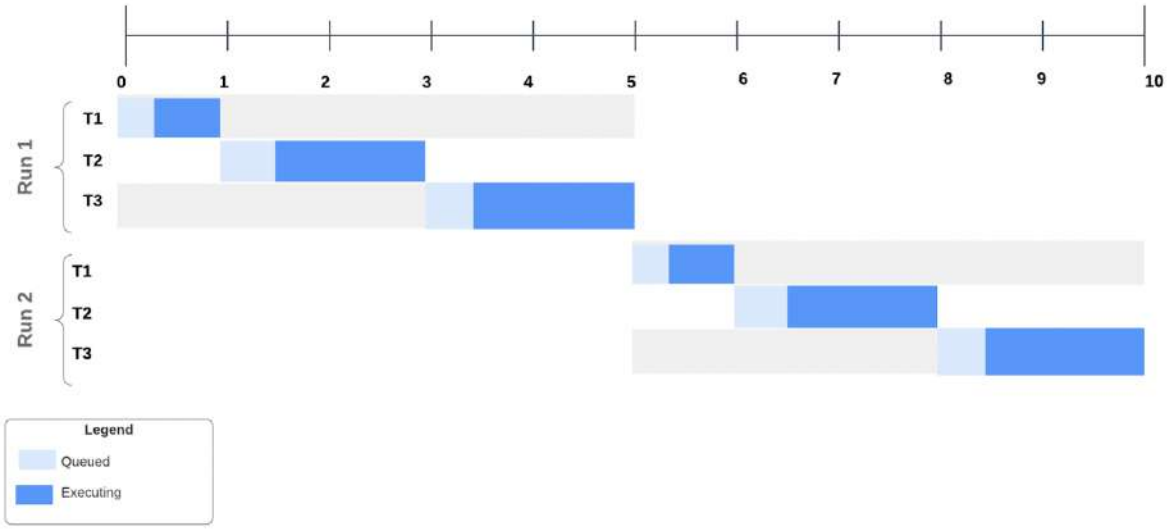
```
when SYSTEM$STREAM_HAS_DATA('strm_applicant1')
as
merge into candidates tgt
using
(select
    id ,
    first_name ,
    last_name ,
    sex ,
    ethinicity ,
    ssn ,
    street_address ,
    str.education_level as ed_level,
    years_of_experience ,
    jdtl.job_id ,
    name ,
    city ,
    state ,
    jdtl.education_level as jreq_level ,
    case when str.education_level = jdtl.education_level then 'SHORTLISTED ' else 'STAGED' end as
status,
    case when str.education_level = jdtl.education_level then 'The education level of candidate matched
with job'
                                        else 'Application recived from candidate'
    end  as comments,
    str.metadata$action,
    str.metadata$isupdate
   from strm_applicant1 str inner join job_details jdtl on (str.job_id = jdtl.job_id)
 ) src
 on tgt.id = src.id
 -- insert clause
 when not matched and src.metadata$action = 'INSERT' and metadata$isupdate = 'FALSE'
 then insert
 values (
    id ,
    first_name ,
    last_name ,
    sex ,
    ethinicity ,
    ssn ,
    street_address,
    ed_level,
    years_of_experience ,
    job_id ,
    name ,
    city ,
    state ,
    jreq_level,
    status,
```

```
   comments
 )
 --update
 when matched and src.metadata$action = 'INSERT' and metadata$isupdate = 'TRUE'
 then update
   set tgt.job_id = src.job_id,
       tgt.candidate_education_level = src.ed_level,
       tgt.required_education_level = src.jreq_level,
       tgt.status = src.status,
       tgt.comments = src.comments
 -- delete
 when matched and src.metadata$action = 'DELETE' and metadata$isupdate = 'FALSE'
 then delete
 ;


-- show task
show tasks ;
--
select * from candidates ;
--
show streams;


alter task tsk_add_candidate_evry_min_use_strm resume ;
show tasks ;
-- Lets insert some data in raw-applicant_staging table
insert into raw_applicant_staging values (111111, 'James',     'Schwartz', 'M',
'American','342-76-9087','5676 Washington Street','High School', 5,10) ;
insert into raw_applicant_staging values (222222, 'Jessica',   'Escobar',  'F',
'Hispanic','456-93-5629','3234 WateringCan Drive','Undergrad', 4, 10) ;

-- check stream
select * from strm_applicant1;
--consume Stream
select * from table(information_schema.task_history()) where completed_time between dateadd(hour,
-1, getdate()) and getdate() order by completed_time desc;
select * from candidates ;

-- insert three more rows and wait a minute
insert into raw_applicant_staging values (333333, 'Ben',       'Hardy',    'M',
'American','876-98-3245','6578 Historic Circle','Masters', 6, 30) ;
insert into raw_applicant_staging values (444444, 'Anjali',     'Singh',   'F', 'Indian
American','435-87-6532','8978 Autumn Day Drive','Masters', 8,20) ;
insert into raw_applicant_staging values (555555, 'Dean',      'Tracy',    'M', 'African','767-34-7656','2343
India Street','Undergrad', 2,50) ;

-- check stream
select * from strm_applicant1;
```

```
--consume Stream
select * from candidates ;

-- Update the data in staging
update raw_applicant_staging set job_id = 10 where id = 555555 ;
-- check stream
select * from strm_applicant1;
--consume Stream
select * from candidates;

-- Pull the records which are inserted
delete from raw_applicant_staging where id = 555555 ;
-- check stream
select * from strm_applicant1;
--consume Stream
select * from candidates ;

show tasks ;

----------


-- cleanup

show streams ;
drop stream if exists STRM_APPLICANT1 ;
show tasks ;
drop task if exists TSK_ADD_CANDIDATE_EVRY_MIN_USE_STRM ;
```

Considerations while estimating a warehouse size when creating a user-managed task

- Average run time of task or DAG = queued time + execution time (shared warehouse)
- Average run time of task or DAG = brief lag + execution time (dedicated warehouse - no queue)
- Task_history = completed time - scheduled time ~ way to get the average run time (que incl)
- Warehouse size : big enough to accommodate more than one child tasks triggered from root task
- Make sure the warehouse is fully utilized by understanding task requirements and selecting the right size.



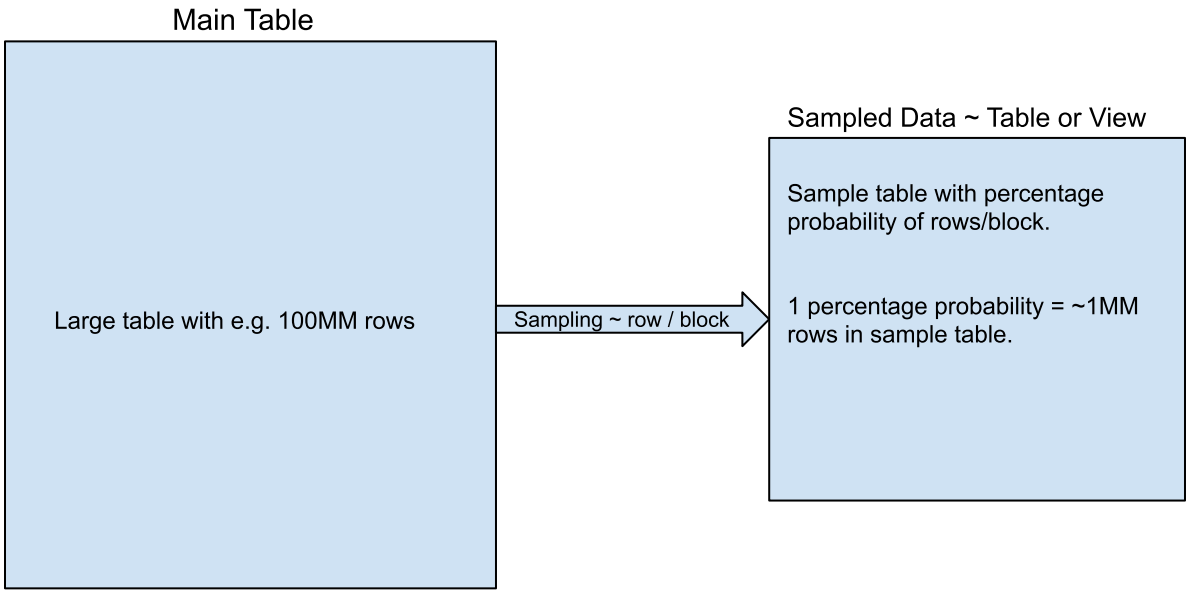
Reference: https://docs.snowflake.com/en/user-guide/tasks-intro.html

SAMPLE / TABLESAMPLE

1. What is sampling data?

   It is very expensive to run and test queries on large table which are big in size such as Terabytes

   So in order to reduce the compute cost, a fraction of data from the large table is sampled into a smaller view or table to get a similar data set as large table.



2. How is sampling done.

Two methods

ROW or BERNOULLI method	SYSTEM or BLOCK
Every row is given a percentage probability to be chosen (0-100)	Each block is given a percentage probability to be chosen (0-100)
Good for small dataset	Good for bigger dataset

Slower	Faster
Support fixed-size sampling (0-1000000)	Doesn’t support fixed-size sampling
Default method	Need to specify

3. SEED / REPEATABLE (0-2147483647) - used to produce the same data set if run again.

4. Syntax

```
SELECT ...
FROM ...
  { SAMPLE | TABLESAMPLE } [ samplingMethod ] ( { <probability> | <num> ROWS } ) [ { REPEATABLE | SEED } ( <seed> ) ]
[ ... ]
```

Where:

```
samplingMethod ::= { { BERNOULLI | ROW } |
                     { SYSTEM | BLOCK } }
```

https://docs.snowflake.com/en/sql-reference/constructs/sample.html

Demo:

-- DATA SAMPLING DEMO --

select count(*) from snowflake_sample_data.tpcds_sf100tcl.customer ; -- 100MM

select c_birth_country, count(*) from snowflake_sample_data.tpcds_sf100tcl.customer group by c_birth_country;

----------Default method with 1 percentage probability ------

```
create or replace table customer_row_1 as
select * from snowflake_sample_data.tpcds_sf100tcl.customer sample (1) seed(1) ; -- takes row if not specifying

create or replace table customer_row_2 as
select * from snowflake_sample_data.tpcds_sf100tcl.customer sample row(1) seed(1) ; --Specify the sampling method

-- check the data
select count(*) from customer_row_1 ; -- 999770
select count(*) from customer_row_2 ; -- 999770

-- Lets prove seed

select * from customer_row_1
minus
select * from customer_row_2
minus
select * from customer_row_1;

create or replace table customer_row_2 as
select * from snowflake_sample_data.tpcds_sf100tcl.customer sample row(1) seed(2) ; --Change the seed number the data will be diff


-- check the data
select count(*) from customer_row_1 ; -- 999770
select count(*) from customer_row_2 ; -- 1,002,056

select * from customer_row_1
minus
select * from customer_row_2
minus
select * from customer_row_1

-- check the sampling
select c_birth_country, count(*) from customer_row_1 group by c_birth_country;
```

---------- percentage probability change from 1 to 25 ----------

```
create or replace table customer_row_3 as
select * from snowflake_sample_data.tpcds_sf100tcl.customer sample row(25) seed(1) ;  -- 25 percentage probability
```


```
-- check data
```

```
select count(*) from customer_row_3 ; -- 25MM rows
select * from customer_row_3  ;

-- check the sampling
select c_birth_country, count(*) from customer_row_3 group by c_birth_country; -- 25 times more
114810/4473
```

----------Fixed Size sampling ----------

```
create or replace table customer_row_4 as
select * from snowflake_sample_data.tpcds_sf100tcl.customer sample row(10000 rows) seed(1) ; --
gives error, seed not supported with fixed.


create or replace table customer_row_4 as
select * from snowflake_sample_data.tpcds_sf100tcl.customer sample row(10000 rows) ;

-- check data
select count(*) from customer_row_4 ; -- 100 rows
select * from customer_row_4  ;

-- check the sampling
select c_birth_country, count(*) from customer_row_4 group by c_birth_country; -- 4473/100 {1M versis
10K}
```

---------- SYSTEM/BLOCK ----------

```
--System/Block method with 1 percentage probability --

create or replace table customer_sys_1 as
select * from snowflake_sample_data.tpcds_sf100tcl.customer sample block(1) seed(1) ; -- system or
block

create or replace table customer_sys_2 as
```

```
select * from snowflake_sample_data.tpcds_sf100tcl.customer sample system(1) seed(1) ; -- system or
block

-- check the data
select count(*) from customer_sys_1 ; -- 1,275,086
select count(*) from customer_sys_2 ; -- 1,275,086

-- Lets prove seed and system or block

select * from customer_sys_1
minus
select * from customer_sys_2
minus
select * from customer_sys_1;

-- recreate table 2 with seed(2)
create or replace table customer_sys_2 as
select * from snowflake_sample_data.tpcds_sf100tcl.customer sample system(1) seed(2) ;


-- check the data
select count(*) from customer_sys_1 ; -- 1,275,086
select count(*) from customer_sys_2 ; -- 1,272,109

select * from customer_sys_1
minus
select * from customer_sys_2
minus
select * from customer_sys_1

-- check the sampling
select c_birth_country, count(*) from customer_sys_1 group by c_birth_country order by 1 ;
select c_birth_country, count(*) from customer_sys_2 group by c_birth_country order by 1 ;




---------- percentage probability change from 1 to 25 ---------------------------------------------

create or replace table customer_sys_3 as
select * from snowflake_sample_data.tpcds_sf100tcl.customer sample system(25) seed(1) ;  -- 25
percentage probability


-- check data
select count(*) from customer_sys_3 ; -- 23MM rows

-- check the sampling
```

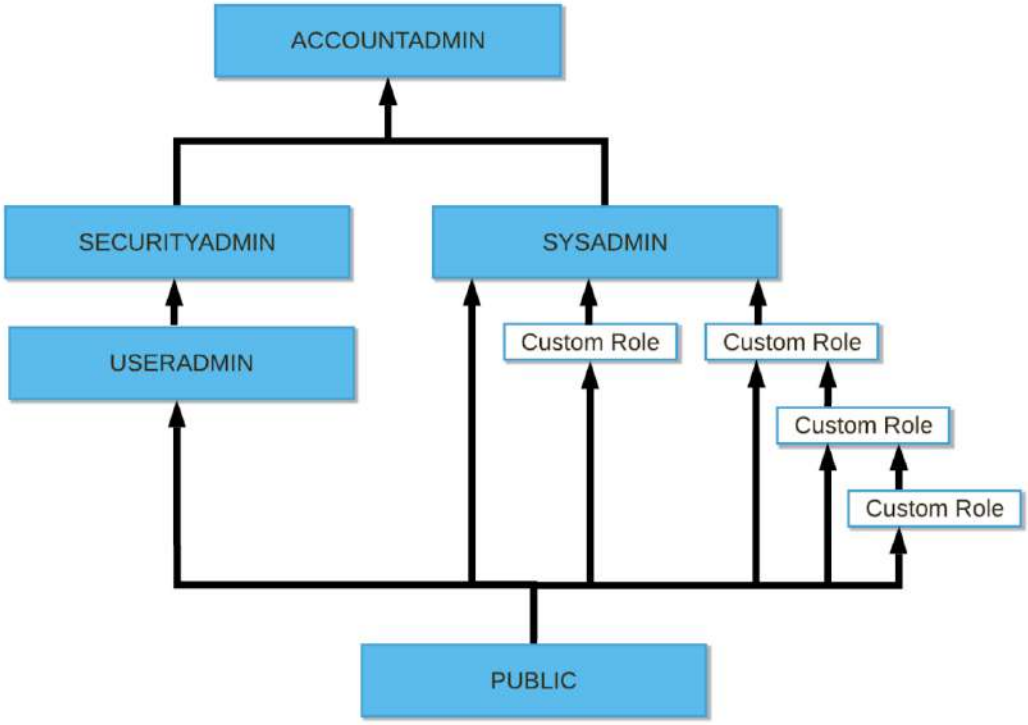
select c_birth_country, count(*) from customer_sys_3 group by c_birth_country order by 1; -- 25 times more ~ 109K

----------Fixed Size sampling ----------
create or replace table customer_sys_4 as
select * from snowflake_sample_data.tpcds_sf100tcl.customer sample system(10000 rows) seed(1) ;
-- seed not supported we have seen in previous session with fixed size sampling

create or replace table customer_sys_4 as
select * from snowflake_sample_data.tpcds_sf100tcl.customer sample system(10000 rows) ; -- block not supported

---

Access management and snowflake objects



---- Access Admin Demo --

show users ;

```
drop user if exists nick_acct_adm ;
drop user if exists dean_sec_adm ;
drop user if exists john_usr_adm ;
drop user if exists jes_sys_adm ;

drop user if exists developer1 ;
drop user if exists developer2 ;
drop user if exists qa1 ;
drop user if exists qa2 ;
drop user if exists reader1 ;
drop user if exists reader2 ;

-- snowflake and vksingh -- which got created at trial

show roles ; -- system defined roles.
drop role if exists developer_role;
drop role if exists qa_role ;
drop role if exists read_role ;


-- ACCOUNTADMIN --

-- As mentioned earlier this is the top level role or most powerful role and should be granted to very
limited people say 2.
   -- It can configure parameters at account level
   -- view billing and manage it
   -- and can terminate the execution of SQL statements any time.
   -- when your account get created the first user is assigned the ACCOUNTADMIN role
      show users ;
-- So lets create one more accountadmin users - Nick. This will make two accountadmin users as
recommended by snowflake

-- create another user having account admin role
create or replace user nick_acct_adm password = '123'
login_name = 'nick_acct_adm'
first_name = 'Nick'
email = 'info@alphaedgesolutions.com'  -- important to give the email for urgent issues if snowflake
wants to connect
default_role = 'ACCOUNTADMIN'
must_change_password = false ;

grant role ACCOUNTADMIN to user nick_acct_adm ;

-- show users
show users ;
show grants to user nick_acct_adm ;

-- MFA authentication should be enabled for accountadmin users
-- Go to Profile and select on enroll MFA
```

```
-- if you want to disable you can run the following command

alter user <username> set DISABLE_MFA = true;



--- SECURITYADMIN — Security administrator – manage object grant globally - modify or revoke any
grant
-- Inherits useradmin
create or replace user dean_sec_adm password = '123'
login_name = 'dean_sec_adm'
first_name = 'Dean'
email = 'info@alphaedgesolutions.com'  -- important to give the email for urgent issues if snowflake
wants to connect
default_role = 'SECURITYADMIN'
must_change_password = false ;

show grants to user dean_sec_adm ;

grant role SECURITYADMIN to user dean_sec_adm ;

-- check grants
show grants to user dean_sec_adm ;


---- USERADMIN ---- users and roles administrator -- dedicated to user and role management --
-- Now we need to create a user having user admin access who can create users and roles

create or replace user john_usr_adm password = '123'
login_name = 'john_usr_adm'
first_name = 'John'
email = 'info@alphaedgesolutions.com'  -- important to give the email for urgent issues if snowflake
wants to connect
default_role = 'USERADMIN'
must_change_password = false ;

grant role USERADMIN to user john_usr_adm ;

show grants to user john_usr_adm ;

---- login --- as user admin to create users and roles --
show users ;


--- SYSADMIN -- create a sysadmin user – system administrator
-- role that has privs to create warehouse/databases and other objects
-- Now we need to create a user having user admin access who can create warehouse, databases and
other objects
```

```
create or replace user jes_sys_adm password = '123'
login_name = 'jes_sys_adm'
first_name = 'Jess'
email = 'info@alphaedgesolutions.com'  -- important to give the email for urgent issues if snowflake
wants to connect
default_role = 'SYSADMIN'
must_change_password = false ;

grant role SYSADMIN to user jes_sys_adm ;

-- show users

show users ;

show grants to user jes_sys_adm ;

-- so all the 4 users are created and we will see how these users control the system and give it to
developers / qa and other users

-- Lets login as john_usr_adm to create some more users who will be using the systems and the related
roles.
```

USERS and ROLES CREATION

—----------

--- USERADMIN—------- user and custom role
-- Login as john_usr_adm/123 and create the users and roles

developer_role

qa_role

read_role

```
create or replace role developer_role ;

create or replace user developer1 password = '123'
login_name = 'developer1'
first_name = 'Developer1'
email = 'info@alphaedgesolutions.com'  -- important to give the email for urgent issues if snowflake
wants to connect
default_role = developer_role
must_change_password = false ;

grant role developer_role to user developer1 ;

show grants to user developer1 ;


create or replace user developer2 password = '123'
login_name = 'developer2'
first_name = 'Developer2'
email = 'info@alphaedgesolutions.com'  -- important to give the email for urgent issues if snowflake
wants to connect
default_role = developer_role
must_change_password = false ;

grant role developer_role to user developer2 ;

show grants to user developer2 ;




------------------------
create or replace role qa_role ;

create or replace user qa1 password = '123'
login_name = 'qa1'
first_name = 'Qa1'
email = 'info@alphaedgesolutions.com'  -- important to give the email for urgent issues if snowflake
wants to connect
default_role = qa_role
must_change_password = false ;

grant role qa_role to user qa1 ;

show grants to user qa1;
```

```
create or replace user qa2 password = '123'
login_name = 'qa2'
first_name = 'Qa2'
email = 'info@alphaedgesolutions.com'  -- important to give the email for urgent issues if snowflake
wants to connect
default_role = qa_role
must_change_password = false ;

grant role qa_role to user qa2 ;

show grants to user qa2 ;



--------------------------
create or replace role read_role ;

create or replace user reader1 password = '123'
login_name = 'reader1'
first_name = 'Reader1'
email = 'info@alphaedgesolutions.com'  -- important to give the email for urgent issues if snowflake
wants to connect
default_role = read_role
must_change_password = false ;

grant role read_role to user reader1 ;

create or replace user reader2 password = '123'
login_name = 'reader2'
first_name = 'Reader2'
email = 'info@alphaedgesolutions.com'  -- important to give the email for urgent issues if snowflake
wants to connect
default_role = read_role
must_change_password = false ;

grant role read_role to user reader2 ;

show grants to user reader1;

show grants to user reader2;

show roles ;
```

—-------SYSADMIN—------CREATE—OBJECTS –DBs–WAREHOUSE —
Login as jes_sys_adm/123

```
create or replace database db_dev ;
create or replace schema stg ;
create or replace schema tgt ;


--
show databases;

use database db_dev ;
use schema stg ;
show schemas ;

--- create warehouse
create or replace warehouse wh_dev ;


-- show
show warehouses ;

show roles ;

-- Create table client_stg
use schema stg ;

create or replace table client_stg
(  id NUMBER(38,0), first_name VARCHAR(16), last_name VARCHAR(50),
   sex VARCHAR(1), ethinicity VARCHAR(30), ssn VARCHAR(15),
   street_address VARCHAR(90),status VARCHAR(10)
);


delete from client_stg ;
```

```
insert into client_stg values (111111, 'James',     'Schwartz', 'M', 'American','342-76-9087','5676
Washington Street','ACTIVE') ;
insert into client_stg values (222222, 'Jessica',   'Escobar',  'F', 'Hispanic','456-93-5629','3234
WateringCan Drive','INACTIVE') ;
insert into client_stg values (333333, 'Ben',       'Hardy',    'M', 'American','876-98-3245','6578 Historic
Circle','INACTIVE') ;
insert into client_stg values (444444, 'Anjali',    'Singh',   'F', 'Indian American','435-87-6532','8978
Autumn Day Drive','ACTIVE') ;
insert into client_stg values (555555, 'Dean',      'Tracy',    'M', 'African','767-34-7656','2343 India
Street','ACTIVE') ;

select * from client_stg ;

-- create a target table

create or replace table client_tgt
(  id NUMBER(38,0), first_name VARCHAR(16), last_name VARCHAR(50),
   sex VARCHAR(1), ethinicity VARCHAR(30), ssn VARCHAR(15),
   street_address VARCHAR(90),status VARCHAR(10)
);


-- create a procedure

  create or replace procedure prc_load_client_tgt()
  returns varchar
  language sql
  as
  $$
    begin
      insert into client_tgt
         select * from client_stg ;

      return 'Record Inserted' ;

   end;
  $$
  ;

-- test the procedure
select * from client_stg ;
select * from client_tgt ; -- empty
call prc_load_client_tgt() ;
select * from client_tgt ; -- procedure works


-- create a view

create or replace view active_client as select * from client_tgt where status = 'ACTIVE' ;
```

```
--
select * from active_client ;

---

show roles ;

show tables ;

show procedures ;

show views ;

grant usage on warehouse wh_dev to developer_role ;
grant usage on database db_dev to developer_role ;
grant usage on schema db_dev.stg to developer_role ;

grant all privileges on database db_dev to developer_role ;
grant all privileges on schema db_dev.stg to developer_role ;

grant select, insert, update, delete on client_stg to developer_role ;
grant select, insert, update, delete on client_tgt to developer_role ;

grant usage on procedure prc_load_client_tgt() to developer_role ;

grant select on view active_client to developer_role ;




--- qa role

grant usage on warehouse wh_dev to qa_role ;
grant usage on database db_dev to qa_role ;
grant usage on schema db_dev.stg to qa_role ;

-- No insert and delete on stg to QA, No insert in tgt
grant select, update on client_stg to qa_role ;
grant select, update, delete on client_tgt to qa_role ;

grant usage on procedure prc_load_client_tgt() to qa_role ;

grant select on view active_client to qa_role ;
```

```
--- read role

grant usage on warehouse wh_dev to read_role ;
grant usage on database db_dev to read_role ;
grant usage on schema db_dev.stg to read_role ;

 -- only select on table
grant select on client_stg to read_role ;
grant select on client_tgt to read_role ;

grant select on view active_client to read_role ;



– developer role should have all access in db_dev on all objects
– qa role should have read and update and delete access in db_qa
– should have read access only


-- drop database (cleanup if needed)
drop database db_dev ;
drop warehouse wh_dev
```

developer1

```
-- check create table and insert access

use schema tgt ;

create or replace table developer1_test_table
(  id NUMBER(38,0), first_name VARCHAR(16), last_name VARCHAR(50),
   sex VARCHAR(1), ethinicity VARCHAR(30), ssn VARCHAR(15),
   street_address VARCHAR(90),status VARCHAR(10)
);

insert into developer1_test_table values (111111, 'James',     'Schwartz', 'M',
'American','342-76-9087','5676 Washington Street','ACTIVE') ;
insert into developer1_test_table values (222222, 'Jessica',   'Escobar',  'F',
'Hispanic','456-93-5629','3234 WateringCan Drive','INACTIVE') ;


select * from developer1_test_table ;


-- existing table insert/update/delete
select * from stg.client_stg ;

select * from client_tgt ;

delete from client_tgt ;

-- execute the procedure
call prc_load_client_tgt() ;

select * from client_tgt ;

select * from active_client ;
```

developer2

```
-- check create table

Use schema tgt ;

show tables ;

create or replace table developer2_test_table
(  id NUMBER(38,0), first_name VARCHAR(16), last_name VARCHAR(50),
   sex VARCHAR(1), ethinicity VARCHAR(30), ssn VARCHAR(15),
   street_address VARCHAR(90),status VARCHAR(10)
);

insert into developer2_test_table values (111111, 'James',     'Schwartz', 'M',
'American','342-76-9087','5676 Washington Street','ACTIVE') ;
insert into developer2_test_table values (222222, 'Jessica',   'Escobar',  'F',
'Hispanic','456-93-5629','3234 WateringCan Drive','INACTIVE') ;


select * from developer2_test_table ;


-- existing table insert/update/delete
select * from stg.client_stg ;

select * from client_tgt ;


delete from client_tgt ;

-- execute the procedure
call prc_load_client_tgt() ;

select * from client_tgt ;
```

```
select * from active_client ;
```

QA1

```
-- check create table

show tables ;

create or replace table qa1_test_table
(  id NUMBER(38,0), first_name VARCHAR(16), last_name VARCHAR(50),
   sex VARCHAR(1), ethinicity VARCHAR(30), ssn VARCHAR(15),
   street_address VARCHAR(90),status VARCHAR(10)
);

-- Table creation not allowed

-- existing table delete not allowed
select * from client_stg ;

delete from client_stg ; -- not allowed

insert into client_stg values (6666666, 'James',     'Schwartz', 'M', 'American','342-76-9087','5676
Washington Street','ACTIVE') ; -- not allowed

-- update

select * from client_stg ;

update client_stg set ethinicity = 'African' where id = 111111 ;

select * from client_stg where id = 111111 ;


delete from client_tgt ;
select * from client_tgt ;

show procedures

-- execute the procedure
```

```
call prc_load_client_tgt() ; -- not allowed

select * from client_tgt ;

show views

select * from active_client ; --


show roles ;

show grants to role qa_role ;
```

— reader1 —

```
-- check create table

show tables ;

create or replace table reader1_test_table
(  id NUMBER(38,0), first_name VARCHAR(16), last_name VARCHAR(50),
   sex VARCHAR(1), ethinicity VARCHAR(30), ssn VARCHAR(15),
   street_address VARCHAR(90),status VARCHAR(10)
);

-- Table creation not allowed

-- existing table delete not allowed
select * from stg.client_stg ;

delete from stg.client_stg ; -- not allowed

insert into stg.client_stg values (6666666, 'James',     'Schwartz', 'M', 'American','342-76-9087','5676
Washington Street','ACTIVE') ; -- not allowed

-- update

select * from stg.client_stg ;

update stg.client_stg set ethinicity = 'African' where id = 111111 ; -- not allowed

select * from client_stg where id = 111111 ;


delete from client_tgt ; -- not allowed
select * from client_tgt ;

show procedures

-- execute the procedure
```

call prc_load_client_tgt() ; -- not allowed

select * from client_tgt ;

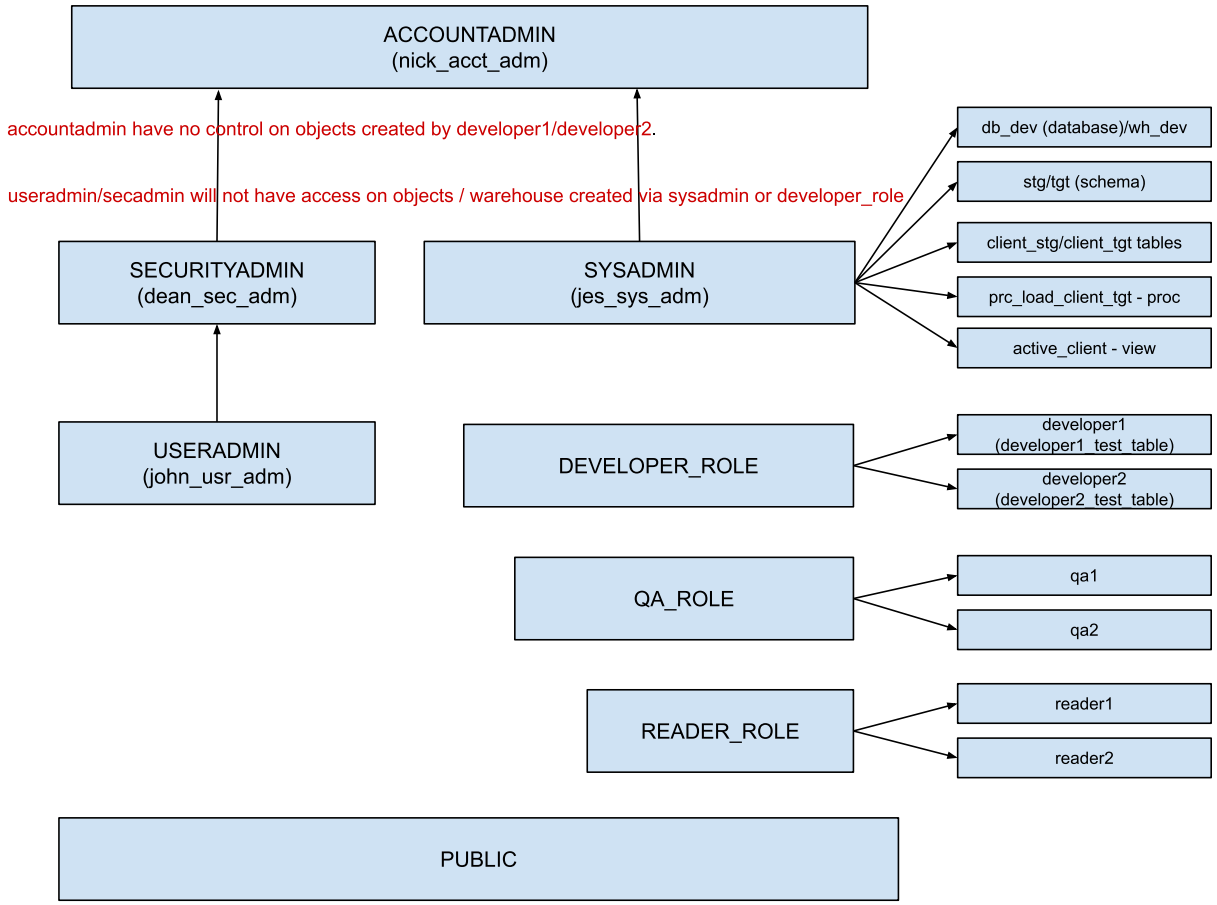
show views

select * from active_client ; --

show roles ;

show grants to role reader_role ;

Two problems:



Problem 1:

NICKACCTADM

```
-- This user have access to DB/Schema/Warehouse since all these were created via sysadmin user
-- and the access is inherited.

show tables in database;

select * from developer1_test_table ; -- No access
select * from developer2_test_table ; -- No access
select * from tgt.client_tgt ; -- allowed  -- since created via sysadmin user
select * from stg.client_stg ; -- allowed -- since created via sysadmin user

drop table developer1_test_table ; -- No access
drop table developer2_test_table ; -- No access

delete from developer1_test_table ; -- No access
delete from developer2_test_table ; -- No access

update developer1_test_table set status = 'INACTIVE' where id = 111111; -- No access

-- insufficient privileges, since the developer role is a custom role and not assigned to sysadmin
```

Problem 2:

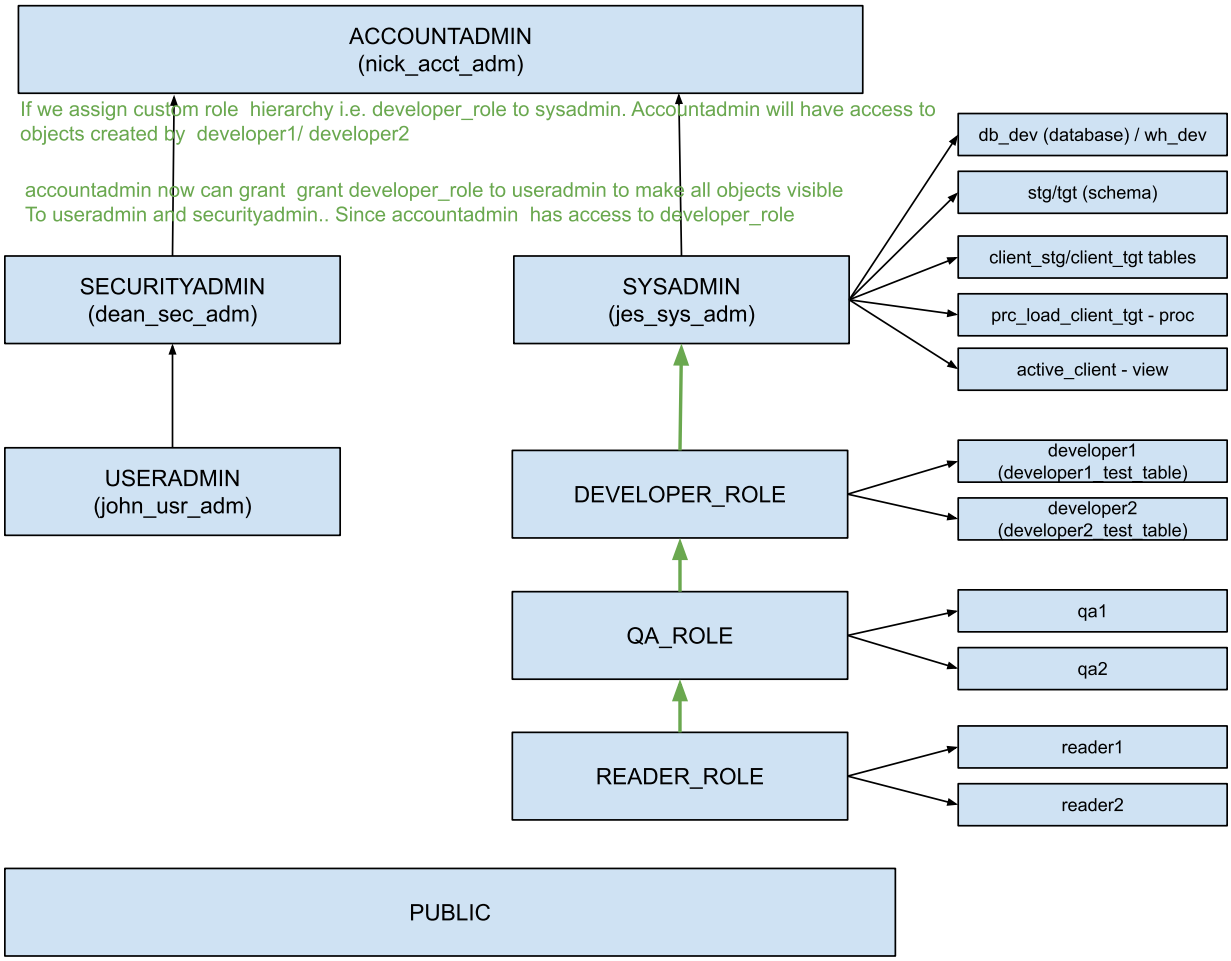
JOHNUSRADM

```
-- No Access to Warehouse
-- No Access to DB
-- No Access to Schemas

show tables in database ; -- No access to any tables
show views in database; -- No access to views
show procedures ; -- No access to procdeure created via sysadmin


-- Can't access the warehouse / DB / Schemas created by sysadmin.
-- sysadmin access is inherited to account admin.
-- Hence the account admin was able to access the warehouse
```

DEANSECADM

```
-- Change the ownership of warehouse compute_wh to use it.
show tables in database ; -- No access to any tables
show views in database; -- No access to views
show procedures ; -- No access to procedure created via sysadmin

-- Manage grants

-- To check manage grants privilegs
grant select on developer1_test_table to securityadmin ;
```

Solutions:



Implement Solution 1:

JOHNUSRADM

```
-- Create hierarchy of roles

-- Grant role <role_name> to role <role>

show roles ;

grant role read_role to role qa_role ;

grant role qa_role to role developer_role ;

grant role developer_role to role sysadmin ;
```

Check Solution 1:

NICKACCTADM

```
show tables in database;

select * from developer1_test_table ;
select * from developer2_test_table ;
select * from tgt.client_tgt ;
select * from stg.client_stg ;
```

```
drop table developer1_test_table ;
drop table developer2_test_table ;

delete from developer1_test_table ;
delete from developer2_test_table ;

select * from developer1_test_table ;

update developer1_test_table set status = 'ACTIVE' where id = 111111;

select * from developer1_test_table ;
```

Implement Solution 2:

NICKACCTADM

```
-- Now accountadmin user can manage and assign the roles. Lot of flexibility now to accountadmin.

grant role developer_role to role useradmin;
```

Check Solution 2 :

JOHNUSRADM

```
-- Now you can select the database/Schema and warehouse

show databases ;
show schemas ;
show tables in database ;
show views in database;
show procedures ;

select * from client_tgt ;
select * from stg.client_stg ;
```

```
select * from developer1_test_table ;
select * from developer2_test_table ;
```

DEANSECADM

```
-- We can see that database / Schema and Warehouse are available now

show tables in database ;
show views in database;
show procedures ;


select * from developer1_test_table ;
select * from developer2_test_table ;
select * from client_tgt ;
select * from stg.client_stg ;

select * from active_client ;
```

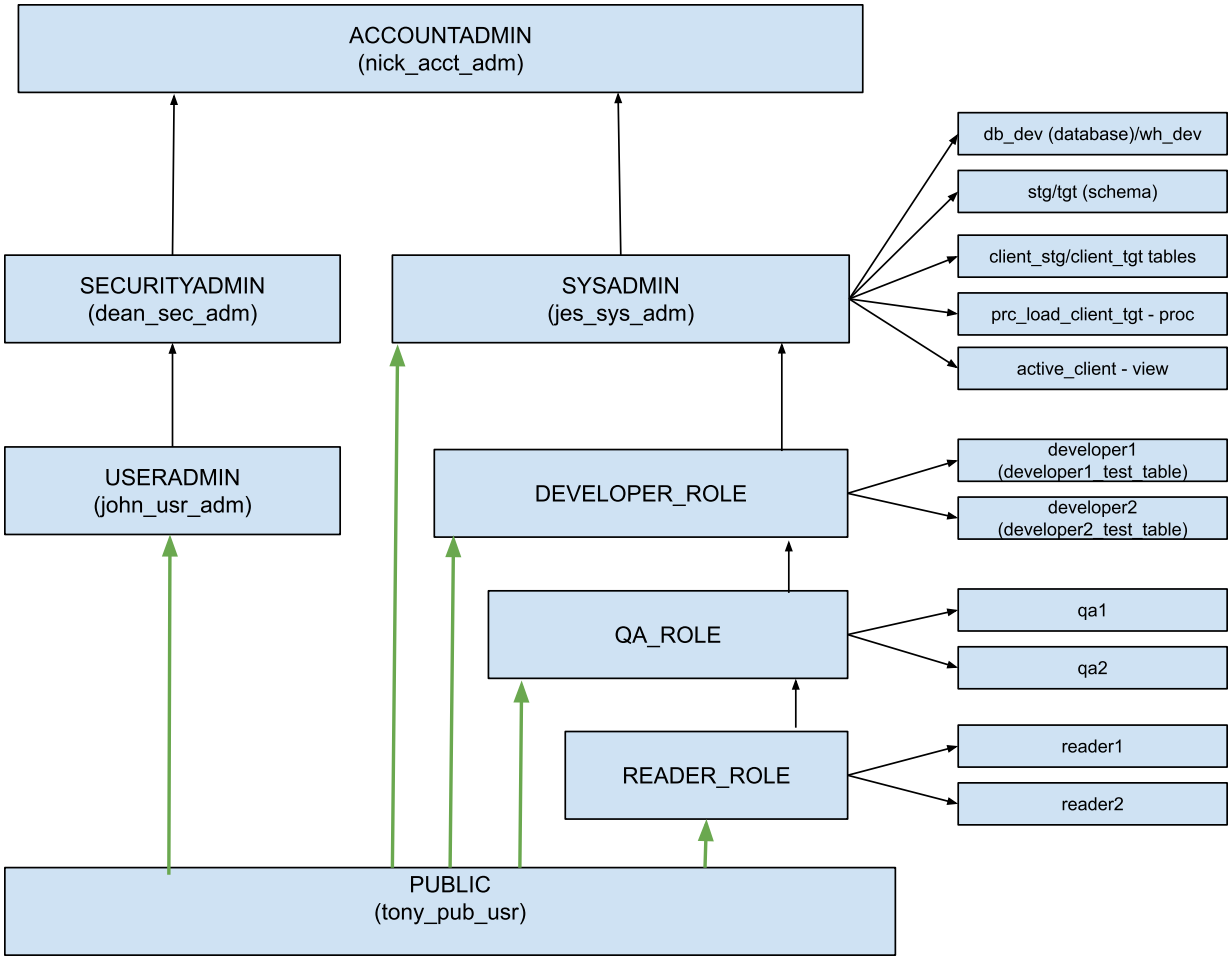
PUBLIC

(tony_pub_usr)

PUBLIC: Pseudo-role that is automatically granted to every user and every role in your account. The PUBLIC role can own securable objects, just like any other role; however, the objects owned by the role are, by definition, available to every other user and role in your account.

This role is typically used in cases where explicit access control is not needed and all users are viewed as equal with regard to their access rights.

Ref: https://docs.snowflake.com/en/user-guide/security-access-control-overview.html



JOHNUSRADM

```
-- create a public user

create or replace user tony_usr_pub password = '123'
login_name = 'tony_usr_pub'
first_name = 'Tony'
email = 'info@alphaedgesolutions.com'  -- important to give the email for urgent issues if snowflake
wants to connect
default_role = public
must_change_password = false ;

-- cleanup

drop user tony_usr_pub ;
```

JESSYSADM

```
create or replace warehouse wh_pub ;
create or replace database db_pub ;
create or replace schema pub ;

grant usage on warehouse wh_pub to public ;
grant usage on database db_pub to public ;
grant usage on schema pub to public ;

grant all privileges on database db_pub to public ;
grant all privileges on schema db_pub.pub to public ;



-- cleanup

drop warehouse if exists wh_pub ;
drop database if exists db_pub ;
```

TONYUSRPUB

```
create or replace table client_pub
(  id NUMBER(38,0), first_name VARCHAR(16), last_name VARCHAR(50),
   sex VARCHAR(1), ethinicity VARCHAR(30), ssn VARCHAR(15),
   street_address VARCHAR(90),status VARCHAR(10)
);


delete from client_pub ;
insert into client_pub values (111111, 'James',     'Schwartz', 'M', 'American','342-76-9087','5676
Washington Street','ACTIVE') ;
insert into client_pub values (222222, 'Jessica',   'Escobar',  'F', 'Hispanic','456-93-5629','3234
WateringCan Drive','INACTIVE') ;
insert into client_pub values (333333, 'Ben',       'Hardy',    'M', 'American','876-98-3245','6578 Historic
Circle','INACTIVE') ;
insert into client_pub values (444444, 'Anjali',    'Singh',   'F', 'Indian American','435-87-6532','8978
Autumn Day Drive','ACTIVE') ;
insert into client_pub values (555555, 'Dean',      'Tracy',    'M', 'African','767-34-7656','2343 India
Street','ACTIVE') ;

select * from client_pub ;
```

Check in other users if this public warehouse / database / schema / table is available to all users. We have not given any explicit grant to any role or user. It is only to public

reader1

```
select * from client_pub ;

delete from client_pub ;
insert into client_pub values (111111, 'James',     'Schwartz', 'M', 'American','342-76-9087','5676 Washington Street','ACTIVE') ;
insert into client_pub values (222222, 'Jessica',   'Escobar',  'F', 'Hispanic','456-93-5629','3234 WateringCan Drive','INACTIVE') ;
insert into client_pub values (333333, 'Ben',       'Hardy',    'M', 'American','876-98-3245','6578 Historic Circle','INACTIVE') ;
insert into client_pub values (444444, 'Anjali',    'Singh',    'F', 'Indian American','435-87-6532','8978 Autumn Day Drive','ACTIVE') ;
insert into client_pub values (555555, 'Dean',      'Tracy',    'M', 'African','767-34-7656','2343 India Street','ACTIVE') ;

select * from client_pub ;

update client_pub set status = 'INACTIVE' where id = 111111;

-- all works
```

deansecadm

```
select * from client_pub ;


delete from client_pub ;

insert into client_pub values (111111, 'James',     'Schwartz', 'M', 'American','342-76-9087','5676 Washington Street','ACTIVE') ;
insert into client_pub values (222222, 'Jessica',   'Escobar',  'F', 'Hispanic','456-93-5629','3234 WateringCan Drive','INACTIVE') ;
```

```
insert into client_pub values (333333, 'Ben',       'Hardy',    'M', 'American','876-98-3245','6578 Historic
Circle','INACTIVE') ;
insert into client_pub values (444444, 'Anjali',     'Singh',   'F', 'Indian American','435-87-6532','8978
Autumn Day Drive','ACTIVE') ;
insert into client_pub values (555555, 'Dean',      'Tracy',    'M', 'African','767-34-7656','2343 India
Street','ACTIVE') ;

select * from client_pub ;

update client_pub set status = 'INACTIVE' where id = 111111;

-- all works
```

nickacctadm

```
select * from client_pub ;


delete from client_pub ;

insert into client_pub values (111111, 'James',     'Schwartz', 'M', 'American','342-76-9087','5676
Washington Street','ACTIVE') ;
insert into client_pub values (222222, 'Jessica',   'Escobar',  'F', 'Hispanic','456-93-5629','3234
WateringCan Drive','INACTIVE') ;
insert into client_pub values (333333, 'Ben',       'Hardy',    'M', 'American','876-98-3245','6578 Historic
Circle','INACTIVE') ;
insert into client_pub values (444444, 'Anjali',     'Singh',   'F', 'Indian American','435-87-6532','8978
Autumn Day Drive','ACTIVE') ;
insert into client_pub values (555555, 'Dean',      'Tracy',    'M', 'African','767-34-7656','2343 India
Street','ACTIVE') ;

select * from client_pub ;

update client_pub set status = 'INACTIVE' where id = 111111;
```

BEST PRACTICES

1. Enable Warehouse Auto Suspend (setting timeout) and Auto Resume

Demo:

Show in UI

```
alter warehouse if exists compute_wh set auto_suspend = 5 ; -- this number is seconds

show warehouses ;

alter warehouse if exists compute_wh set auto_suspend = 12 ; -- this number is seconds

show warehouses ;

alter warehouse if exists compute_wh set auto_suspend = 600; -- this number is seconds, hence 10 minutes

show warehouses ;

alter warehouse if exists compute_wh set auto_resume = false;

show warehouses ;

alter warehouse if exists compute_wh set auto_resume = true;

show warehouses ;
```

Auto-Suspend - based on the need - can be 0, 10 or 5 as needed,

2. Monitor workload and usage metrics
    a. Long running queries
    b. Failed queries

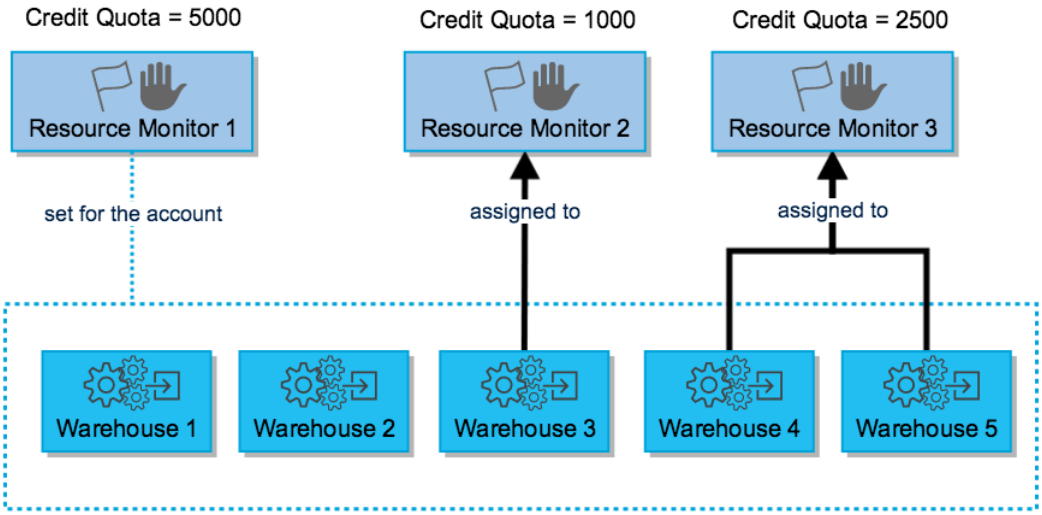
```
-- set timeout

show parameters like '%TIMEOUT%' ;

--STATEMENT_QUEUED_TIMEOUT_IN_SECONDS  -> 0 (disabled) if non 0, statement will be canceled if more seconds passed

--STATEMENT_TIMEOUT_IN_SECONDS -> 0 means (604800) ~ 168 hrs, otherwise give the seconds after which the execution should be canceled.

-- can be set at session and warehouse level. Let's set it up at warehouse level.
--
alter warehouse compute_wh set STATEMENT_TIMEOUT_IN_SECONDS = 3600 ;

-- warehouse
show parameters like '%TIMEOUT%' in warehouse compute_wh;

-- session
show parameters like '%TIMEOUT%' in session ;
```

```
-- account usage -- storage usage
select * from snowflake.account_usage.table_storage_metrics ;
--
-- long running queries
select * from snowflake.account_usage.query_history where execution_status = 'SUCCESS' order by total_elapsed_time desc; -- time elapsed is in milliseconds

-- Failed Queries
select * from snowflake.account_usage.query_history where execution_status = 'FAIL' order by total_elapsed_time desc; -- time elapsed is in milliseconds
```

```
-- warehouse metering history -- compute power
select * from snowflake.account_usage.warehouse_metering_history ;

-- you ca use various grouping in SQL to generate the credits used

-- lets see how much credits are used by compute_wh in last 15 days
select warehouse_name, sum(credits_used) from
snowflake.account_usage.warehouse_metering_history
where start_time >= dateadd(day,-15,current_timestamp()) group by warehouse_name order by
2 desc ;

-- multiple combinations can be tried
```

3. Resource Monitors - Very important to control the credit usage from compute cost in your account.

   Compute cost can be - warehouse + 10% cloud services



Reference: https://docs.snowflake.com/en/user-guide/resource-monitors.html

Demo:

UI

+

SQL.

```
---- RESOURCE MONITORS-----
-- see the resource monitors
show resource monitors ;
-- drop command to drop a resource monitor
drop resource monitor if exists RSRC_MNTR_WH;
-- alter or modifying a resource monitor.
alter resource monitor rsrc_mntr_account set credit_quota  = 10 ;


-- create a resource monitor default schedule -- only accountadmin can create it --

-- Step 1 : create resource monitor
-- Step 2 : assign warehouse or assign account to resource monitor

--step1
create or replace resource monitor rsrc_mntr_wh with credit_quota = 10 triggers on 90 percent do
suspend ;

--step2
alter warehouse compute_wh set resource_monitor = rsrc_mntr_wh ;

alter account set resource_monitor = rsrc_mntr_wh ; -- error

-- solutions
alter warehouse compute_wh set resource_monitor = null ;
alter account set resource_monitor = rsrc_mntr_wh ; -- success



--- create a resource monitor as custom schedule
--step1
create or replace resource monitor rsrc_mntr_cstm with credit_quota = 10
frequency = monthly
start_timestamp = immediately
triggers on 90 percent do suspend ;

--step2
alter warehouse compute_wh set resource_monitor = rsrc_mntr_cstm ;


--- more notifications - upto 5 notification can be given

--step1
```

```
create or replace resource monitor rsrc_mntr_cstm2 with credit_quota = 10
frequency = monthly
start_timestamp = immediately
triggers
        on 70 percent do notify
        on 70 percent do suspend
        on 80 percent do suspend_immediate;

--step2
alter warehouse compute_wh set resource_monitor = rsrc_mntr_cstm2 ;

--- error for more than 5 notify

create or replace resource monitor rsrc_mntr_cstm3 with credit_quota = 10
frequency = monthly
start_timestamp = immediately
triggers
        on 70 percent do notify
        on 71 percent do notify
        on 72 percent do notify
        on 73 percent do notify
        on 74 percent do notify
        on 70 percent do suspend
        on 80 percent do suspend_immediate;

alter warehouse compute_wh set resource_monitor = rsrc_mntr_cstm3 ;


create or replace resource monitor rsrc_mntr_cstm4 with credit_quota = 10
frequency = monthly
start_timestamp = immediately
triggers
        on 70 percent do notify
        on 71 percent do notify
        on 72 percent do notify
        on 73 percent do notify
        on 74 percent do notify
        on 75 percent do notify
        on 70 percent do suspend
        on 80 percent do suspend_immediate; -- error only 5 are allowed

alter warehouse compute_wh set resource_monitor = rsrc_mntr_cstm4 ;
```

4. Small warehouse in Non Prod

   DEV and SIT have less workloads so a smaller warehouse is sufficient.

   Warehouses are not needed to be multi-cluster in Non Prod as workload is less.

5. Don't select * all columns - Specific columns

If you select specific columns it will minimize the amount of data to be transferred and read.

AVOID: select * from table
USE: select col1, col5, col10 from table.

Select only what is needed,

6. Check historic patterns of query and benchmark

As mentioned earlier - keep scanning the query history and understand the queries to Benchmark.

Keep checking, if benchmark exceeds check the query plan, warehouse size and clustering Strategies and see if some changes are required.

7. Reduce Storage cost

   Check table if they are not used for long time and purge

   Time travel should be set carefully.

   Try using temporary tables for development purpose where the results are needed in session Only.

   See the usage of transient tables and try to use transient instead of permanent tables by default.

THANK YOU